THÈSE

présentée

pour l'obtention

du

DIPLOME de DOCTEUR de 3e CYCLE

à

L'UNIVERSITÉ PIERRE et MARIE CURIE

(Paris 6e)

Spécialité : MATHÉMATIQUES

Mention : INFORMATIQUE

par M. Jean-Pierre FAYE

Sujet de la thèse :

DES LOGICIELS EFFICACES DE CALCUL ET DE CONTROLE DES VALEURS PROPRES ET VECTEURS PROPRES

soutenue le 18 mai 1982 devant la commission de :

M. J. ARSAC, *Président*

M. J. VIGNES, *Examinateur*

M. C. BREZINSKÍ, *Examinateur*

M. J. ABADIE, *Examinateur*

M. R. ALT, *Examinateur*

M. A. QUIGNODON, *Invité*

UNIV. LIBRARY

THÈSE

présentée

pour l'obtention

du

DIPLOME de DOCTEUR de 3e CYCLE

à

L'UNIVERSITÉ PIERRE et MARIE CURIE
(Paris 6e)

Spécialité : MATHÉMATIQUES

Mention : INFORMATIQUE

par M. Jean-Pierre FAYE

Sujet de la thèse :

DES LOGICIELS EFFICACES DE CALCUL ET DE CONTROLE DES VALEURS PROPRES ET VECTEURS PROPRES

soutenue le 18 mai 1982 devant la commission de :

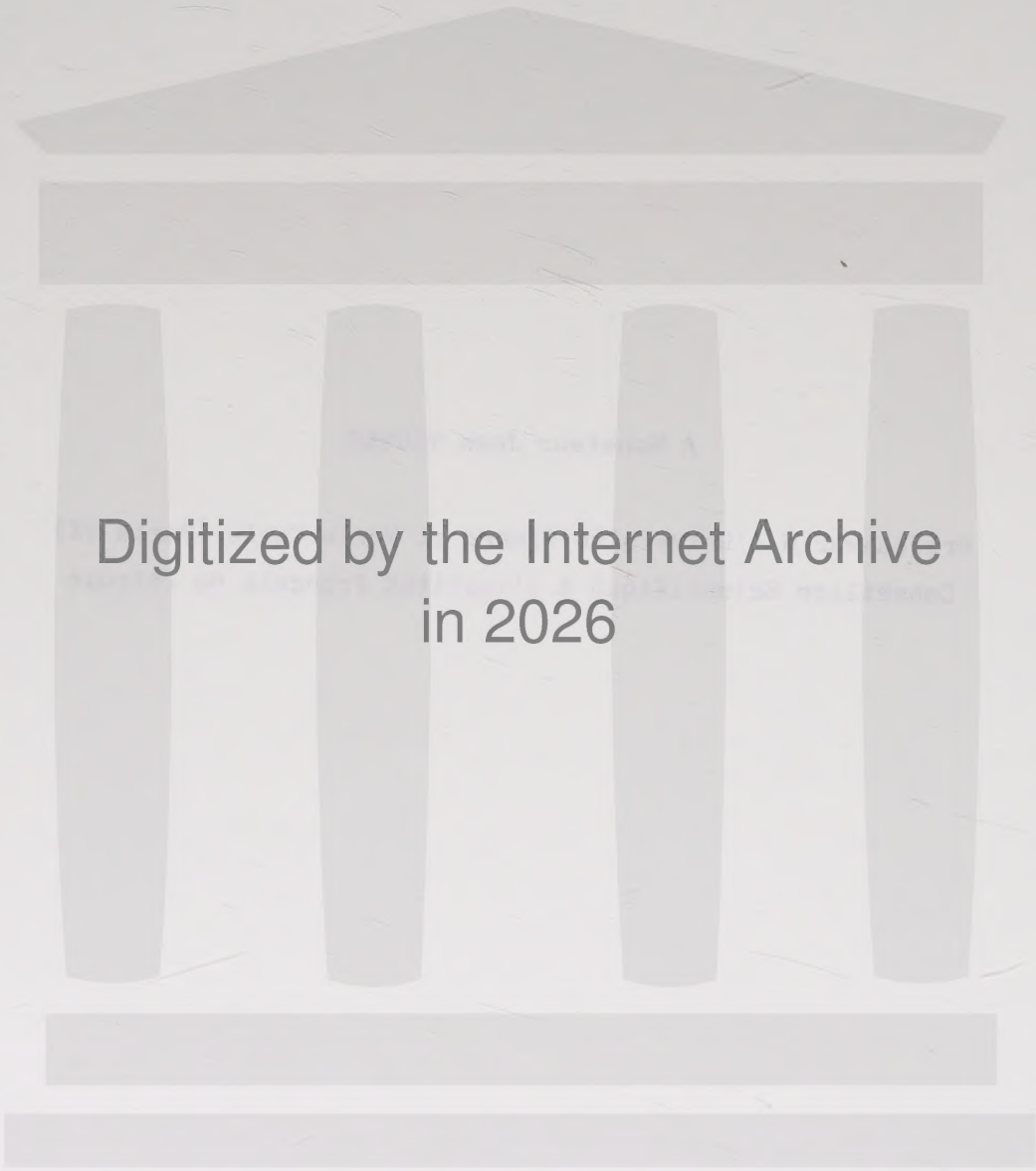
M. J. ARSAC, *Président*
M. J. VIGNES, *Examinateur*
M. C. BREZINSKÍ, *Examinateur*
M. J. ABADIE, *Examinateur*
M. R. ALT, *Examinateur*
M. A. QUIGNODON, *Invité*

A mes Parents

A ma Femme

A Monsieur Jean VIGNES,

Professeur à l'Université Pierre et Marie Curie (Paris VI)
Conseiller Scientifique à l'Institut Français du Pétrole


Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41543523_0024

Je tiens à remercier Monsieur le Professeur J. ARSAC qui a bien voulu me faire l'honneur de présider le Jury de cette thèse.

A Monsieur le Professeur J. VIGNES, qui a inspiré et dirigé mes travaux, je lui adresse mes sentiments de profonde gratitude.

Je remercie Monsieur le Professeur C. BREZINSKI ainsi que Monsieur le Professeur J. ABADIE d'avoir accepté de participer à ce Jury.

J'en remercie également Monsieur R. ALT, Maître-assistant à l'Université Pierre et Marie Curie, en témoignage des conseils qu'il m'a prodigués.

Mes remerciements vont aussi à Monsieur A. QUIGNODON, de la Société Franlab-Informatique (Groupe CISI) pour l'intérêt qu'il a bien voulu porter à ce travail.

Je suis reconnaissant à la Direction de la Compagnie Internationale de Services en Informatique et à Franlab de m'avoir fourni les moyens de conduire ces travaux à bonne fin.

TABLE DES MATIERES

I - INTRODUCTION 1

II - STABILITE DE L'ALGORITHME QR 5

II.1 - L'algorithme QR usuel 5

II.1.1 - Principe 5
II.1.2 - Mise en oeuvre de l'algorithme 8
II.1.3 - Tests d'arrêt usuels 9
II.1.4 - Notice d'utilisation et organigramme simplifié du programme AQRCT 10
II.1.5 - Programme FORTRAN AQRCT 13
II.1.6 - Résultats numériques 25

II.2 - Tests d'arrêt optimaux 29

II.2.1 - Nécessité d'un bon test d'arrêt 29
II.2.2 - Expression des tests d'arrêt optimaux 29
II.2.3 - Problème de mise en oeuvre 30

II.3 - Méthode de perturbation 31

II.3.1 - Principe 31
II.3.2 - Mise en oeuvre de la méthode de perturbation pour l'application d'un test d'arrêt optimal 34
II.3.3 - Programmes utilitaires 35
II.3.4 - Notice d'utilisation du programme CHSIQR ... 37
II.3.5 - Programme FORTRAN CHSIQR (1ère version) 39
II.3.6 - Résultats numériques 52

II.4 - Méthode de multiplication 55

II.4.1 - Principe 55
II.4.2 - Mise en oeuvre de la méthode de multiplication pour l'application d'un test d'arrêt optimal 57
II.4.3 - Programme FORTRAN CHSIQR (2ème version) ... 60
II.4.4 - Résultats numériques 74

II.5 - Conclusions 76

III - RESOLUTION DU POLYNOME CARACTERISTIQUE 77

III.1 - Introduction 77

III.2 - La méthode de Faddeev 77

III.2.1 - Principe 77
III.2.2 - Utilisation de la méthode de multiplication. 79
III.2.3 - Mise en oeuvre 80

III.2.4 - Notice d'utilisation et organigramme simplifié du programme POLCAR 80
III.2.5 - Programme FORTRAN POLCAR 83
III.2.6 - Résultats numériques 87

III.3 - Résolution du polynome calculé 89

III.3.1 - Utilisation de la méthode de Bairstow stabilisée 89
III.3.2 - Résultats numériques 89

III.4 - Conclusions 91

IV - AMELIORATION DES VALEURS PROPRES ET DES VECTEURS PROPRES 93

IV.1 - Introduction 93

IV.2 - Amélioration des valeurs propres 93

IV.2.1 - Recherche de la singularité numérique 93
IV.2.2 - Utilisation du programme OPTIM de J. Vignes .. 94
IV.2.3 - Principe du programme OPTIM modifié 94
IV.2.4 - Convergence du problème 99
IV.2.5 - Détection de la singularité numérique par la méthode de permutation 100
IV.2.6 - Mise en oeuvre 104
IV.2.7 - Notice d'utilisation des programmes AOPTIM et COPTIM 109
IV.2.8 - Programmes FORTRAN AOPTIM et COPTIM 110
IV.2.9 - Résultats numériques 118
IV.2.10 - Utilisation du polynôme caractéristique 124
IV.2.11 - Notice d'utilisation du programme AMONIT 128
IV.2.12 - Programme FORTRAN AMONIT 129
IV.2.13 - Résultats numériques 133

IV.3 - Amélioration des vecteurs propres 134

IV.3.1 - Notion de résidu minimal 134
IV.3.2 - Une méthode de minimisation sous contrainte .. 138
IV.3.3 - Convergence de la méthode 145
IV.3.4 - Détection des zéros numériques par la méthode de multiplication 147
IV.3.5 - Mise en oeuvre 148
IV.3.6 - Notice d'utilisation du programme MINF2 152
IV.3.7 - Programme FORTRAN MINF2 153
IV.3.8 - Résultats numériques 159

IV.4 - Conclusions 166

V - CONCLUSIONS GENERALES ET EXTENSIONS 167

Bibliographie 169

I - INTRODUCTION

De par la représentation discontinue des nombres en machine, les résultats mathématiques et les résultats informatiques des algorithmes ne peuvent être toujours identiques.

De plus, les algorithmes utilisés sont souvent des résultats de théorèmes, eux-mêmes seulement vérifiés pour des espaces continus du type $\mathbb{R}^n$ ou $\mathbb{C}^n$. Cette continuité sur les espaces de travail est une hypothèse essentielle, par exemple, pour les théorèmes de convergence. En conséquence, une incertitude sur la validité des résultats numériques obtenus en machine reste forte quelle que soit la précision de l'ordinateur utilisé.

La précision des résultats obtenus sera fonction de deux critères :

1) La stabilité de l'algorithme (susceptibilité de créer des erreurs numériques).

2) Le conditionnement du problème (variation du résultat en fonction de petites variations des données).

Nous appellerons signal le résultat théorique d'un algorithme (obtenu en précision illimitée). Nous appellerons bruit l'erreur numérique associée à un résultat machine (précision limitée). Ce bruit numérique est l'effet cumulé des erreurs successives (arrondi ou troncature) introduites au cours des calculs, et de leur propagation |37, 39|.

Nous justifions cette terminologie par le fait que l'erreur numérique sur le résultat se comporte comme une variable aléatoire Gaussienne centrée dans le cas général |25|.

Dans ce cadre, les algorithmes de calcul de valeurs propres et de vecteurs propres n'offrent qu'une stabilité relative aux erreurs d'arrondi ou de troncature et ce, pour un problème pouvant être mal conditionné |13, 14, 18, 39|.

Plusieurs algorithmes de calcul de valeurs propres existent : méthodes de JACOBI, GIVENS-HOUSEHOLDER, LR, QR, Puissance itérée, résolution du polynome caractéristique, etc.

Une des méthodes les plus stables et des plus économiques en nombre d'opérations restent la méthode QR |9|. C'est également la méthode la plus utilisée |33|. Nous allons donc nous attacher à étudier l'effet du bruit numérique sur cet algorithme, et ce dans le cas d'une matrice réelle quelconque |24|.

Nous ne sommes d'ailleurs pas les premiers à étudier la stabilité des valeurs propres au bruit numérique.

D'une manière générale, si λ_1 est valeur propre de A, la valeur propre calculée, baptisée $\lambda_1(E)$, est considérée comme valeur propre de (A+E) où E est une matrice de bruit issue de la propagation des erreurs d'arrondi ou de troncature.

Dans |18, 39| nous trouvons des estimations de la norme de Frobenius de E (notée $\|E\|_E$). Cette norme est définie par :

$$\|A\|_E = \sqrt{\sum_{i,j} |A_{i,j}|^2} = \sqrt{\text{trace } (A^*.A)}$$

où A* matrice auto-adjointe de A

$\text{trace } M = \sum_i M_{i,i}$

Wilkinson donne alors dans |40| une mesure de la sensibilité d'une valeur propre aux variations pour une matrice A ; son conditionnement cond(λ) avec cond(λ) $\in [1,+\infty[$.

Cette valeur peut être exprimée de la façon suivante : si x est vecteur propre droit de A, $Ax = \lambda x$, si y* est vecteur propre gauche de A, $y^*A = \lambda y^*$.

Alors : $\text{Cond}(\lambda) = \|y\|_2 \cdot \|x\|_2 / |y^*.x|$, où $\|y\|_2 = \sqrt{(y^*.y)}$.

Enfin, en utilisant la théorie des perturbations |19| et pour $\|E\|_E$ suffisamment petite, de l'équation :

$$(A+E)\, x_1(E) = \lambda_1(E) x_1(E)$$

le développement de la perturbation donne :

$$|\lambda_1 - \lambda_1(E)| \leq \text{cond}(\lambda_1).\|E\|_E$$

Il est donc possible de calculer une borne d'erreurs sur les valeurs propres obtenues a posteriori |5,6|.

L'inconvénient de telles méthodes est que l'estimation de $|\lambda_1 - \lambda_1(E)|$ reste finalement très imprécise. De plus, elles ne permettent pas un contrôle en cours de calcul.

Nous allons donc partir sur une autre voie et utiliser la méthode de permutation-perturbation introduite par Vignes |37|. Nous verrons que l'introduction de cette méthode à l'intérieur de l'algorithme QR permet de bien séparer signal et bruit, et ce à chaque étape du calcul. Cette dernière possibilité nous permettra, en plus d'une bonne estimation de l'erreur finale sur les valeurs propres, de mettre en place des tests d'arrêt optimaux au cours du calcul.

Nous proposerons ensuite une nouvelle méthode d'évaluation du bruit numérique que nous appellerons méthode de multiplication (nous verrons pourquoi). Bien que cette méthode soit moins générale que la précédente, nous verrons que pour l'algorithme QR elle est plus aisée à mettre en oeuvre, plus économique, et donne des résultats similaires, ce qui nous permettra de la justifier a posteriori.

Une autre façon de calculer les valeurs propres est de calculer le polynôme caractéristique, puis de le résoudre. Cette méthode avait été abandonnée à cause de son instabilité. Les travaux récents de R. Alt et J. Vignes ont permis de stabiliser la méthode de Bairstow pour la résolution des équations algébriques |38, Tome 2|.

Nous proposons donc une mise en oeuvre du calcul du polynôme caractéristique par la méthode Faddeev |8| avec évaluation de la précision sur les coefficients du polynôme par la méthode de multiplication.

Nous pensons que le plus grand défaut d'un algorithme n'est pas son instabilité, mais l'inconnu sur la précision des résultats. Sur un exemple précis, nous montrons que la résolution du polynôme caractéristique peut donner de meilleurs résultats que l'algorithme Q.R. Mais l'essentiel est bien de pouvoir estimer que ces résultats sont meilleurs.

Les valeurs propres étant calculées, leurs précisions connues, le problème de leur amélioration reste entier dans le cas d'une précision insuffisante.

Nous pouvons les recalculer en double précision, ou même en précision étendue. Mais l'augmentation de la précision a ses limites.

Il existe des programmes d'amélioration de la valeur propre, mais, dans le cas de valeurs propres mal conditionnées, nous ne pouvons être certain de la stabilité de l'algorithme d'amélioration de la valeur propre.

Dans la 4ème partie, nous proposerons donc un algorithme d'amélioration de la valeur propre pour lequel nous sommes sûr que la valeur propre a bien été améliorée. Nous nous inspirerons largement des travaux de J. Vignes, à la fois pour tester la singularité d'une matrice par la méthode des permutations |23 et 31|, et à la fois pour l'algorithme de recherche lui-même, dans le cas où la valeur propre calculée est complexe |36|. Nous verrons, hélas, que les améliorations sont dépendantes de la stabilité de l'algorithme de calcul du déterminant, et que la méthode de Gauss généralement utilisée est très instable. Dans l'attente d'un algorithme de calcul du déterminant plus stable, nous proposons une amélioration à partir du calcul du polynôme caractéristique.

Enfin, si le calcul des valeurs propres pose un problème, celui des vecteurs propres associés en pose bien d'autres !

Après triangularisation de la matrice initiale, toujours par la méthode QR avec conservation des transformations, nous résolvons aisément un système d'équations par remontée du système, puis nous effectuons une transformation inverse sur les vecteurs solutions. La précision des vecteurs propres est donc dépendante de celle des valeurs propres.

Si nous effectuons une amélioration des valeurs propres, il faut donc pouvoir ensuite améliorer les vecteurs propres associés. Nous nous y emploierons dans la dernière partie en développant une méthode originale et en utilisant de nouveau la méthode de multiplication comme moyen de contrôle du bruit numérique.

Dans la conclusion, nous verrons que ces méthodes sont immédiatement applicables au problème de la valeur propre généralisée |2, 11, 20|.

II - STABILITE DE L'ALGORITHME QR

II.1 - L'ALGORITHME QR USUEL

II.1.1 - Principe

La méthode QR due à Francis |9| et à Kublanovskaya |21| est une amélioration de l'algorithme LR |18, 32|. Nous allons en rappeler les grandes lignes pour finalement arriver à la version définie dans |41|.

Nous rappelons que toute matrice carrée A d'ordre n à éléments complexes peut être exprimée comme le produit QR d'une matrice unitaire Q et d'une matrice triangulaire supérieure R |32|.

Si A est inversible, cette factorisation est unique si l'on convient de choisir les éléments diagonaux de R réels et positifs.

L'algorithme QR consiste, en posant $A_1 = A$, à engendrer une suite A_s de matrices unitairement semblables à la matrice A. A la s-ième étape, nous factorisons la matrice A_s par :

$$A_s = Q_s R_s \tag{2.1.1}$$

Si Q* désigne la matrice adjointe de Q :

$$Q_s^* A_s Q_s = R_s Q_s \tag{2.1.2}$$

Si nous définissons :

$$A_{s+1} = R_s Q_s \tag{2.1.3}$$

alors A_{s+1} est bien semblable à A_s et les itérés successifs satisfont aux équations :

$$A_{s+1} = Q_s^* A_s Q_s = Q_s^* Q_{s-1}^* \cdots Q_1^* A_1 Q_1 \cdots Q_{s-1} Q_s \tag{2.1.4}$$

On montre (B. Parlett |28|, J.H. Wilkinson |40|) qu'en arithmétique complexe la matrice A_s, semblable à A, converge vers une matrice triangulaire supérieure.

Des soucis de stabilité et d'économie vont alors aboutir à l'algorithme QR avec translation de l'origine et étape double, permettant de travailler en arithmétique réelle si A est réelle |24|.

La matrice A a été mise préalablement sous forme Hessenberg supérieure |16| par une méthode directe |41|.

L'algorithme QR avec translation de l'origine est décrit par les relations :

$$\begin{cases} Q_s(A_s - k_s I) = R_s \\ A_{s+1} = R_s\, Q_s^T + k_s I \end{cases} \qquad (2.1.5)$$

où A_s, Q_s, R_s sont réelles, I est la matrice unité et k_s la translation de l'origine.

Nous obtenons encore : $A_{s+1} = Q_s\, A_s\, Q_s^T$ (2.1.6)

Nous pouvons montrer que si A_1 a la forme de Hessenberg supérieure, alors A_s a la même forme quel que soit s. Parlett |29| montre aussi que si tous les k_s tendent vers 0, alors en général A_s tend vers une forme telle que :

$$a_{i+1,i}^{(s)}\; a_{i+2,i+1}^{(s)} = 0 \qquad (2.1.7)$$

(où les coefficients $a_{i,j}^{(s)}$ sont les termes de la matrice A^s).

Alors les valeurs propres peuvent être isolées sur la diagonale ou bien sont les valeurs propres des matrices diagonales 2x2 restantes.

L'étape double |9| permet d'effectuer deux étapes QR avec deux valeurs réelles de k_s et k_{s+1} ou avec des valeurs complexes conjuguées, sans utiliser l'arithmétique complexe.

Soit :

$$A_{s+2} = Q_{s+1}\, Q_s\, A_s\, Q_s^T\, Q_{s+1}^T \qquad (2.1.8)$$

donnant :

$$A_s(Q_s^T Q_{s+1}^T) = (Q_s^T\, Q_{s+1}^T)\, A_{s+2} \qquad (2.1.9)$$

$$(Q_s^T\, Q_{s+1}^T)(R_{s+1}\, R_s) = (A_s - k_s I)(A_s - k_{s+1} I) \qquad (2.1.10)$$

Soit alors : $$\begin{cases} Q_{s+1}\, Q_s = Q \;;\; R_{s+1}\, R_s = R \\ (A_s - k_s I)(A_s - k_{s+1} I) = M \end{cases} \qquad (2.1.11)$$

Nous avons :

$$A_s Q^T = Q^T A_{s+2} \quad \text{et} \quad R = QM \tag{2.1.12}$$

Les translations de l'origine effectuées à chaque étape sont prises comme les racines du dernier bloc 2x2 diagonal de la matrice As. Soit :

$$\begin{cases} k_s + k_{s+1} = a^{(s)}_{n-1,n-1} + a^{(s)}_{n,n} \\ k_s\, k_{s+1} = a^{(s)}_{n-1,n-1}\, a^{(s)}_{n,n} - a^{(s)}_{n-1,n}\, a^{(s)}_{n,n-1} \end{cases} \tag{2.1.13}$$

Nous pouvons montrer alors que Q (2.1.11) est réelle et qu'elle peut être identifiée à un produit de matrices Hermitiennes élémentaires |15, 17| $P_1, P_2, \ldots, P_{n-1}$ où :

$$P_r = I - 2\,\omega_r\,\omega_r^t \tag{2.1.14}$$

et où ω_r est un vecteur unité avec des zéros comme (r-1) premières composantes.

Nous utilisons un ω_r^t de la forme :

$$(0,\ldots,0,\alpha_r,\ \beta_r,\ \gamma_r,0,\ldots 0) \tag{2.1.15}$$

α_r, β_r, γ_r étant en position r, r+1 et r+2 respectivement, et déterminés par les éléments (r,r-1),(r+1,r-1),(r+2,r-1) de la matrice courante :

$$P_{r-1} \ldots P_1\, A_s\, P_1^t \ldots P_{r-1}^t \tag{2.1.16}$$

La matrice P_1 est seulement déterminée par la lère colonne de M (2.1.11) qui est de la forme :

$$(p_1,\ q_1,\ r_1,\ 0,\ \ldots\ 0)^t \tag{2.1.17}$$

avec :
$$\begin{cases} p_1 = a^{(s)2}_{1,1} - a^{(s)}_{1,1}\,(k_s + k_{s+1}) + k_s\, k_{s+1} + a^{(s)}_{1,2}\, a^{(s)}_{2,1} \\ q_1 = a^{(s)}_{2,1}\,(a^{(s)}_{1,1} + a^{(s)}_{2,2} - k_s - k_{s+1}) \\ r_1 = a^{(s)}_{3,1}\, a^{(s)}_{2,1} \end{cases} \tag{2.1.18}$$

De même, les éléments p_m, q_m, r_m seront définis par :

$$\begin{cases} p_m = a_{m,m}^{(s)^2} - a_{m,m}^{(s)} (k_s + k_{s+1}) + k_s\, k_{s+1} + a_{m,m+1}^{(s)}\, a_{m+1,m}^{(s)} \\ q_m = a_{m+1,m}^{(s)} (a_{m,m}^{(s)} + a_{m+1,m+1}^{(s)} - k_s - k_{s+1}) \\ r_m = a_{m+2,\, m+1}^{(s)}\, a_{m+1,m}^{(s)} \end{cases} \qquad (2.1.19)$$

r_m n'étant pas calculé pour P_{n-1}, donc pour ω_{n-1}.

Enfin, à la fois par souci de réduction de volume de calcul et de stabilité, $2\,\omega_s\,\omega_s^t$ est exprimé par la forme $u_s.v_s^t$, où u_s et v_s sont parallèles à ω_s.

Les éléments non nuls sont alors donnés respectivement par :

$$((p_s \pm \sigma_s) / (\pm \sigma_s),\ q_s / (\pm \sigma_s),\ r_s / (\pm \sigma_s)) \qquad (2.1.20)$$

et $$(1 \quad ,\ q_s/(p_s \pm \sigma_s),\ r_s/(p_s \pm \sigma_s)) \qquad (2.1.21)$$

avec $$\sigma_s^2 = p_s^2 + q_s^2 + r_s^2 \qquad (2.1.22)$$

Nous retrouverons cette description dans |24|.

II.1.2 - Mise en oeuvre de l'algorithme

Nous dirons qu'une étape est franchie lorsque les termes $a_{n,n-1}^{(s)}$ ou $a_{n-1,n-2}^{(s)}$ auront été trouvés négligeables. Nous définirons dans le paragraphe suivant ce dernier terme.

Si $a_{n,n-1}^{(s)}$ est négligeable, on effectue une déflation d'ordre 1 ; si c'est $a_{n-1,n-2}^{(s)}$, on effectue une déflation d'ordre 2.

Nous rappelons qu'une déflation d'ordre p est un abaissement de l'ordre n de la matrice initiale A à l'ordre n-p.

L'algorithme sera terminé quand, par déflations successives, nous serons ramenés à un ordre 1 ou 2. Ces déflations utilisées dans |24| permettent une accélération de la convergence.

Un premier test d'arrêt est donc nécessaire après chaque itération sur les coefficients sous-diagonaux et même avant chaque étape pour éviter les itérations inutiles.

Nous appellerons un pas le passage de :

$$P_{r-1} \ldots P_1 A_s P_1^t \ldots P_{r-1}^t \qquad (2.1.23)$$

à

$$P_r P_{r-1} \ldots P_1 A_s P_1^t \ldots P_{r-1}^t P_r^t \qquad (2.1.24)$$

Au pas r, la matrice courante se différencie d'une matrice Hessenberg supérieure uniquement par les éléments en position (r+1, r-1) et (r+2, r-1). Il est clair que si ces éléments sont négligeables, les pas 1 à r peuvent être évités, car les termes de plus haut rang nous intéressent seuls pour une convergence rapide sur les coefficients $a^{(s)}_{n,n-1}$ et $a^{(s)}_{n-1,n-2}$ pour une matrice d'ordre n à l'étape courante.

Un deuxième test sera donc effectué selon |24| afin d'éviter les pas inutiles.

Afin d'obtenir les valeurs propres par modules décroissants, à chaque étape, les valeurs propres de l'étape précédente seront comparées et une permutation éventuelle sera effectuée sur les blocs 2x2, 3x3 ou 4x4 selon les cas de figure. Dans les cas de bloc 3x3 ou 4x4, l'algorithme QR sera encore utilisé après permutation.

Une mise en forme de Hessenberg supérieure préalable de la matrice A est donc nécessaire pour cet algorithme. Nous en trouverons la description dans |41|.

Enfin, si T est la matrice des transformations de mise en forme de Hessenberg supérieure, il est aisé de calculer simultanément l'algorithme QR et les transformations :

$$Q_s \ldots Q_1 . T \qquad (2.1.25)$$

à partir des coefficients p_s, q_s et r_s.

II.1.3 - Tests d'arrêt usuels

Comme dans tout algorithme itératif, le délicat problème de tester des quantités négligeables se pose. De plus, ici, la précision des valeurs propres calculées en fin d'algorithme va être fonction des approximations successives que nous aurons faites sur les premières valeurs propres calculées.

Il est donc nécessaire que les approximations effectuées soient minimales.

Le test d'arrêt utilisé dans |41| sur les termes sous-diagonaux de A est de la forme :

$$|a_{\ell,\ell-1}| \leq \varepsilon * (|a_{\ell-1,\ell-1}| + |a_{\ell,\ell}|) \qquad (2.1.26)$$

Wilkinson préconise de choisir pour ε la plus petite valeur positive pour laquelle :

$$1 + \varepsilon \neq 1 \quad \text{en machine}$$

Cette constante dépendante de la machine est appelée la quantité macheps.

Les termes diagonaux utilisés dans ce test ne sont que des ordres de grandeur de référence.

Celui utilisé afin d'éviter les pas inutiles est de la forme :

$$|a_{m,m-1}|(|q_m|+|r_m|) \leq \varepsilon\ |p_m|(|a_{m+1,m+1}|+|a_{m,m}|+|a_{m-1,m-1}|) \qquad (2.1.27)$$

Il ne faut voir en ce test qu'une évaluation grossière des coefficients sous-diagonaux en position (r+1,r-1) et (r+2,r-1) afin de le simplifier (d'après les auteurs).

Ce sont donc ces deux tests qui seront appliqués sur CDC 7600 avec $\varepsilon = 2^{-47}$ dans le programme mis en oeuvre, baptisé AQRCT.

Par ailleurs, si, au bout de 20 itérations, aucune valeur propre n'a été trouvée, nous effectuons une translation spéciale de l'origine. Nous recommençons à 40 itérations et nous arrêtons l'algorithme à 60 itérations.

II.1.4 - Notice d'utilisation et organigramme simplifié du programme AQRCT

Objet

Calcul des valeurs propres par ordre de module décroissant par la méthode QR d'une matrice Hessenberg supérieure, avec conservation des transformations.

Appel

```
CALL AQRCT(A,N,IA,INF,ISUP,EPS,VP,ITYPE,VECT)
```

Arguments d'entrée

A — Bloc réel bidimensionné dont la première dimension est IA et la seconde au moins égale à N, contenant les éléments de la matrice Hessenberg supérieure.

$$A(I,J) = a_{ij} \qquad i,j = 1,\ldots,N$$

A est détruite au cours du calcul.

N — entier, ordre de la matrice (a_{ij}).

IA — première dimension des blocs A,VP,ITYPE,VECT

INF,ISUP — Les calculs ne seront effectués que sur le bloc diagonal (a_{ij}) avec i,j = INF,INF+1,...,ISUP. On a :

$$1 \leqslant INF < ISUP \leqslant N$$

EPS — précision relative demandée sur le test des coefficients sous-diagonaux. Valeurs conseillées :

2^{-47} sur CDC 7600 (48 bits de mantisse)

Arguments de sortie

VP — Bloc réel bidimensionné dont la 1ère dimension est IA et la seconde 2.
VP(I,1) et VP(I,2) contiennent respectivement les parties réelles et imaginaires des N valeurs propres λ_i calculées.

ITYPE — Bloc entier de dimensions au moins égales à N contenant -1,0,1 ou 2

-1 valeur propre non trouvée
0 valeur propre réelle
1 valeur propre complexe à partie imaginaire positive
2 valeur propre complexe à partie imaginaire négative

VECT — Bloc réel bidimensionné dont la 1ère dimension est IA et la seconde au moins égale à N. Contient la matrice des transformations. Cette matrice doit être préalablement initialisée à la matrice identité si une transformation préalable n'a pas été nécessaire pour rendre la matrice A Hessenberg supérieure. Dans le cas contraire, elle doit contenir la matrice des transformations issue de la mise en forme Hessenberg supérieure.

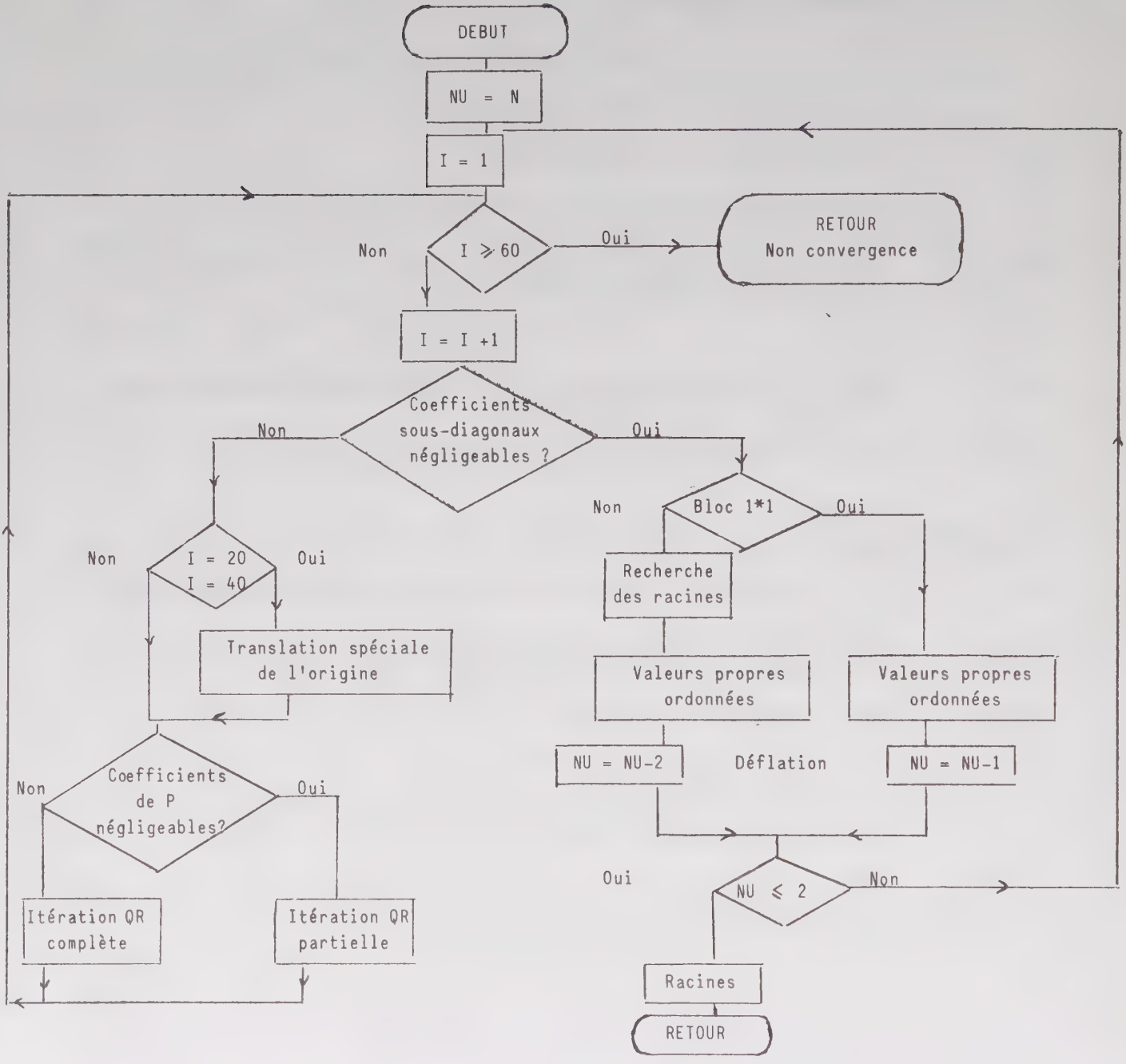
Sous-programmes employés

QR — Sous-programme effectuant une itération QR.

ECHANGE — Sous-programme de permutation des blocs pour des valeurs propres ordonnées.

RACINE — Sous-programme de calcul des valeurs propres à partir des blocs résiduels.

Organigramme simplifié



II.1.5 - Programme FORTRAN AQRCT

```
      SUBROUTINE AQRCT(A,N,MAXL,INF,ISUP,EPS,VP,ITYPE,VECT)                00000300
      DIMENSION A(MAXL,MAXL),VP(MAXL,2),VECT(MAXL,MAXL),ITYPE(MAXL)        00000310
      LOGICAL FAIL                                                         00000320
C                                                                          00000330
C                                                                          00000340
C     INITIALISATION                                                       00000350
C                                                                          00000360
      IF(INF.EQ.1)GOTO 1111                                                00000370
      INF1=INF-1                                                           00000380
      DO 1101 K=1,INF1                                                     00000390
      ITYPE(K)=0                                                           00000400
      VP(K,1)=A(K,K)                                                       00000410
      VP(K,2)=0.                                                           00000420
      DO 1102 J=1,N                                                        00000430
 1102 VECT(J,K)=0.                                                         00000440
 1101 VECT(K,K)=1.                                                         00000450
 1111 IF(ISUP.EQ.N)GOTO 2222                                               00000460
      ISUP1=ISUP+1                                                         00000470
      DO 1103 K=ISUP1,N                                                    00000480
      VP(K,1)= A(K,K)                                                      00000490
      VP(K,2)=0.                                                           00000500
      DO 1144 J=1,N                                                        00000510
 1144 VECT(J,K)=0.                                                         00000520
 1103 VECT(K,K)=1.                                                         00000530
C                                                                          00000540
 2222 CONTINUE                                                             00000550
      DO 10 I=INF,ISUP                                                     00000560
      ITYPE(I)=-1                                                          00000570
   10 CONTINUE                                                             00000580
      T=0.                                                                 00000590
C                                                                          00000600
C     RECHERCHE ET MISE EN ORDRE DES VALEURS PROPRES                       00000610
C                                                                          00000620
      NU=ISUP                                                              00000630
  100 IF(NU.LT.INF)GOTO 500                                                00000640
      IT=0                                                                 00000650
C                                                                          00000660
C     ELEMENTS SOUS-DIAG NEGLIGEABLES                                      00000670
C                                                                          00000680
  110 CONTINUE                                                             00000690
      L=NU                                                                 00000700
  120 CONTINUE                                                             00000710
      IF(L.EQ.INF)GOTO 130                                                 00000720
      IF(ABS(A(L,L-1)).LT.EPS*(ABS(A(L-1,L-1))+ABS(A(L,L))))GOTO 130       00000730
      L=L-1                                                                00000740
      GOTO 120                                                             00000750
  130 CONTINUE                                                             00000760
C                                                                          00000770
C     EST-CE UN BLOC 1*1 OU 2*2 QUI SE DETACHE                             00000780
C                                                                          00000790
      X=A(NU,NU)                                                           00000800
      IF(L.EQ.NU)GOTO 300                                                  00000810
      Y=A(NU-1,NU-1)                                                       00000820
      W=A(NU,NU-1)*A(NU-1,NU)                                              00000830
      IF(L.EQ.NU-1)GOTO 200                                                00000840
C                                                                          00000850
C     SI PLUS DE 20 ITERATIONS ALORS TRANSLATION SPECIALE                  00000860
```

```
      C                                                                 00000870
            IF(IT.EQ.60)GOTO 500                                        00000880
60          IF(IT.NE.20.AND.IT.NE.40)GOTO 150                           00000890
      C                                                                 00000900
      C     TRANSLATION                                                 00000910
      C                                                                 00000920
            T=T+X                                                       00000930
65          DO 140 I=INF,NU                                             00000940
            A(I,I)=A(I,I)-X                                             00000950
       140  CONTINUE                                                    00000960
            S=ABS(A(NU,NU-1))+ABS(A(NU-1,NU-2))                         00000970
            X=0.75*S                                                    00000980
70          Y=X                                                         00000990
            W=-0.4375*S**2                                              00001000
       150  CONTINUE                                                    00001010
            IT=IT+1                                                     00001020
      C                                                                 00001030
75    C     RECHERCHE DE DEUX ELEMENTS SOUS-DIAGONAUX NEGLIGEABLES      00001040
      C                                                                 00001050
            NL=NU-2                                                     00001060
       160  CONTINUE                                                    00001070
            Z=A(NL,NL)                                                  00001080
80          R=X-Z                                                       00001090
            S=Y-Z                                                       00001100
            P=(R*S-W)/A(NL+1,NL)+A(NL,NL+1)                             00001110
            Q=A(NL+1,NL+1)-Z-R-S                                        00001120
            R=A(NL+2,NL+1)                                              00001130
85          S=ABS(P)+ABS(Q)+ABS(R)                                      00001140
            P=P/S                                                       00001150
            Q=Q/S                                                       00001160
            R=R/S                                                       00001170
            IF(NL.EQ.L)GOTO 170                                         00001180
90          IF(ABS(A(NL,NL-1))*(ABS(Q)+ABS(R)).LE.                      00001190
           1EPS*ABS(P)*(ABS(A(NL-1,NL-1))+ABS(Z)+ABS(A(NL+1,NL+1))))GOTO 170 00001200
            NL=NL-1                                                     00001210
            GOTO 160                                                    00001220
       170  CONTINUE                                                    00001230
95    C     ITERATION QR ENTRE NL ET NU                                 00001240
      C                                                                 00001250
      C                                                                 00001260
            CALL QR(A,N,MAXL,P,Q,R,NL,NU,VECT)                          00001270
            GOTO 110                                                    00001280
100   C                                                                 00001290
      C     UN BLOC 2*2                                                 00001300
      C                                                                 00001310
       200  IF(NU.NE.INF+1)A(NU-1,NU-2)=0.                              00001320
            A(NU,NU)=A(NU,NU)+T                                         00001330
105         A(NU-1,NU-1)=A(NU-1,NU-1)+T                                 00001340
            ITYPE(NU)=0                                                 00001350
            ITYPE(NU-1)=0                                               00001360
            NU=NU                                                       00001370
       210  CONTINUE                                                    00001380
110         NL=NU-1                                                     00001390
      C                                                                 00001400
      C     BLOC 2*2 ,VALEUR PROPRES REELLES ?                          00001410
      C                                                                 00001420
            CALL RACINE(A,N,MAXL,NL,E1,E2,VECT)                         00001430
```

```
      C
      C     SI OUI  ON STOCK CES DEUX VALEURS                                  00001440
      C                                                                        00001450
      C                                                                        00001460
            IF(A(MU,MU-1).EQ.0.)GOTO 310                                       00001470
            IF(MU.EQ.ISUP)GOTO 400                                             00001480
            IF(MU.EQ.ISUP-1)GOTO 220                                           00001490
            IF(A(MU+2,MU+1).EQ.0.)GOTO 220                                     00001500
      C                                                                        00001510
      C     LE PROCHAIN BLOC EST 2*2                                           00001520
      C                                                                        00001530
            IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)                     00001540
           1  .GE.A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*A(MU+2,MU+1))GOTO 40000001550
            CALL ECHANGE(A,N,MAXL,NL,2,2,EPS,FAIL,VECT)                        00001560
            IF(.NOT.FAIL)GOTO 215                                              00001570
            ITYPE(NL)=-1                                                       00001580
            ITYPE(NL+1)=-1                                                     00001590
            ITYPE(NL+2)=-1                                                     00001600
            ITYPE(NL+3)=-1                                                     00001610
            GOTO 500                                                           00001620
       215  CONTINUE                                                          00001630
            MU=MU+2                                                            00001640
            GOTO 230                                                           00001650
       220  CONTINUE                                                          00001660
      C                                                                        00001670
      C     BLOC 1*1                                                           00001680
      C                                                                        00001690
            IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)                     00001700
           1 .GE.A(MU+1,MU+1)**2)GOTO 400                                      00001710
            CALL ECHANGE(A,N,MAXL,NL,2,1,EPS,FAIL,VECT)                        00001720
            IF(.NOT.FAIL)GOTO 225                                              00001730
            ITYPE(NL)=-1                                                       00001740
            ITYPE(NL+1)=-1                                                     00001750
            ITYPE(NL+2)=-1                                                     00001760
            GOTO 500                                                           00001770
       225  CONTINUE                                                          00001780
            MU=MU+1                                                            00001790
       230  CONTINUE                                                          00001800
            GOTO 210                                                           00001810
      C                                                                        00001820
      C     UNE VALEUR PROPRE A ETE TROUVEE                                    00001830
      C                                                                        00001840
       300  NL=0                                                              00001850
            A(NU,NU)=A(NU,NU)+T                                                00001860
            IF(NU.NE.INF)A(NU,NU-1)=0.                                         00001870
            ITYPE(NU)=0                                                        00001880
            MU=NU                                                              00001890
      C                                                                        00001900
       310  CONTINUE                                                          00001910
       320  CONTINUE                                                          00001920
            IF(MU.EQ.ISUP)GOTO 350                                             00001930
            IF(MU.EQ.ISUP-1)GOTO 330                                           00001940
            IF(A(MU+2,MU+1).EQ.0.)GOTO 330                                     00001950
            IF(A(MU,MU)**2.GE.A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)           00001960
           1  *A(MU+2,MU+1))GOTO 400                                           00001970
            CALL ECHANGE(A,N,MAXL,MU,1,2,EPS,FAIL,VECT)                        00001980
            IF(.NOT.FAIL)GOTO 325                                              00001990
            ITYPE(MU)=-1                                                       00002000
```

```
                ITYPE(MU+1)=-1                                                  00002010
                ITYPE(MU+2)=-1                                                  00002020
                GOTO 500                                                        00002030
175          325 CONTINUE                                                       00002040
                MU=MU+2                                                         00002050
                GOTO 340                                                        00002060
             330 CONTINUE                                                       00002070
             C                                                                  00002080
180          C     BLOC 1*1                                                     00002090
             C                                                                  00002100
                IF(ABS(A(MU,MU)).GE.ABS(A(MU+1,MU+1))) GOTO 350                 00002110
                CALL ECHANGE(A,N,MAXL,MU,1,1,EPS,FAIL,VECT)                     00002120
                MU=MU+1                                                         00002130
185          340 CONTINUE                                                       00002140
                GOTO 320                                                        00002150
             350 CONTINUE                                                       00002160
                MU=NL                                                           00002170
                NL=0                                                            00002180
190             IF(MU.NE.0)GOTO 310                                             00002190
             400 CONTINUE                                                       00002200
                NU=L-1                                                          00002210
                GOTO 100                                                        00002220
             C                                                                  00002230
195          C     TOUTES LES V.P. ONT ETE TROUVEES ET ORDONNEES                00002240
             C                                                                  00002250
             500 IF(NU.LT.INF)GOTO 507                                          00002260
                DO 503 I=1,NU                                                   00002270
                A(I,I)=A(I,I)+T                                                 00002280
200          503 CONTINUE                                                       00002290
             507 CONTINUE                                                       00002300
                NU=ISUP                                                         00002310
             510 CONTINUE                                                       00002320
                IF(ITYPE(NU).NE.-1)GOTO 515                                     00002330
205             NU=NU-1                                                         00002340
                GOTO 540                                                        00002350
             515 CONTINUE                                                       00002360
                IF(NU.EQ.INF)GOTO 520                                           00002370
                IF(A(NU,NU-1).EQ.0.)GOTO 520                                    00002380
210          C                                                                  00002390
             C     BLOC 2*2                                                     00002400
             C                                                                  00002410
                NUM1=NU-1                                                       00002420
                CALL RACINE(A,N,MAXL,NUM1,E1,E2,VECT)                           00002430
215             IF(A(NU,NU-1).EQ.0.)GOTO 520                                    00002440
                VP(NU,1)=E1                                                     00002450
                VP(NU-1,2)=E2                                                   00002460
                VP(NU-1,1)=VP(NU,1)                                             00002470
                VP(NU,2)=-VP(NU-1,2)                                            00002480
220             ITYPE(NU-1)=1                                                   00002490
                ITYPE(NU)=2                                                     00002500
                NU=NU-2                                                         00002510
                GOTO 530                                                        00002520
             520 CONTINUE                                                       00002530
225          C                                                                  00002540
             C     RACINE SIMPLE                                                00002550
             C                                                                  00002560
                VP(NU,1)=A(NU,NU)                                               00002570
```

```
                    VP(NU,2)=0.                                          00002580
230                 NU=NU-1                                              00002590
                530 CONTINUE                                             00002600
                540 CONTINUE                                             00002610
                    IF(NU.GE.INF)GOTO 510                                00002620
                    RETURN                                               00002630
235                 END                                                  00002640
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 AQRCT

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	EPS	REAL		F.P.
705	E1	REAL			706	E2	REAL		
661	FAIL	LOGICAL			666	I	INTEGER		
0	INF	INTEGER		F.P.	662	INF1	INTEGER		
0	ISUP	INTEGER		F.P.	665	ISUP1	INTEGER		
671	IT	INTEGER			0	ITYPE	INTEGER	ARRAY	F.P.
664	J	INTEGER			663	K	INTEGER		
672	L	INTEGER			0	MAXL	INTEGER		F.P.
704	MU	INTEGER			0	N	INTEGER		F.P.
677	NL	INTEGER			670	NU	INTEGER		
707	NUM1	INTEGER			702	P	REAL		
703	Q	REAL			701	R	REAL		
676	S	REAL			667	T	REAL		
0	VECT	REAL	ARRAY	F.P.	0	VP	REAL	ARRAY	F.P.
675	W	REAL			673	X	REAL		
674	Y	REAL			700	Z	REAL		

EXTERNALS	TYPE	ARGS			
ECHANGE		9	QR		9
RACINE		7			

INLINE FUNCTIONS	TYPE	ARGS	
ABS	REAL	1	INTRIN

STATEMENT LABELS

0	10	105	100	110	110		
115	120	126	130	0	140		
170	150	201	160	235	170		
245	200	270	210	327	215		
333	220	357	225	362	230		
363	300	375	310	375	320		
424	325	426	330	444	340		
445	350	450	400	453	500		
0	503	465	507	472	510		
500	515	531	520	540	530		
540	540	0	1101	0	1102		
0	1103	37	1111	0	1144		
70	2222						

```
      SUBROUTINE ECHANGE(A,N,MAXL,L,IB1,IB2,EPS,FAIL,VECT)                00002650
      DIMENSION A(MAXL,MAXL),VECT(MAXL,MAXL)                              00002660
C$    ARRAYS                                                              00002670
      LOGICAL FAIL                                                        00002680
      FAIL=.FALSE.                                                        00002690
      IF(IB1.EQ.2)GOTO 40                                                 00002700
      IF(IB2.EQ.2)GOTO 10                                                 00002710
C                                                                         00002720
C     ECHANGE DE DEUX BLOC 1*1                                            00002730
C                                                                         00002740
      L1=L+1                                                              00002750
      Q=A(L+1,L+1)-A(L,L)                                                 00002760
      P=A(L,L+1)                                                          00002770
      R=AMAX1(P,Q)                                                        00002780
      IF(R.EQ.0.)RETURN                                                   00002790
      P=P/R                                                               00002800
      Q=Q/R                                                               00002810
      R=SQRT(P**2+Q**2)                                                   00002820
      P=P/R                                                               00002830
      Q=Q/R                                                               00002840
      DO 3 J=L,N                                                          00002850
      S=P*A(L,J)+Q*A(L+1,J)                                               00002860
      A(L+1,J)=P*A(L+1,J)-Q*A(L,J)                                        00002870
      A(L,J)=S                                                            00002880
    3 CONTINUE                                                            00002890
      DO 5 I=1,L1                                                         00002900
      S=P*A(I,L)+Q*A(I,L+1)                                               00002910
      A(I,L+1)=P*A(I,L+1)-Q*A(I,L)                                        00002920
      A(I,L)=S                                                            00002930
    5 CONTINUE                                                            00002940
      DO 7 I=1,N                                                          00002950
      S=P*VECT(I,L)+Q*VECT(I,L+1)                                         00002960
      VECT(I,L+1)=P*VECT(I,L+1)-Q*VECT(I,L)                               00002970
      VECT(I,L)=S                                                         00002980
    7 CONTINUE                                                            00002990
      A(L+1,L)=0.                                                         00003000
      RETURN                                                              00003010
   10 CONTINUE                                                            00003020
C                                                                         00003030
C     ECHANGE 1*1 2*2                                                     00003040
C                                                                         00003050
      X=A(L,L)                                                            00003060
      P=1.                                                                00003070
      Q=1.                                                                00003080
      R=1.                                                                00003090
      LK2=L+2                                                             00003100
      CALL QR(A,N,MAXL,P,Q,R,L,LK2,VECT)                                  00003110
      IT=0                                                                00003120
   20 IT=IT+1                                                             00003130
      IF(IT.LE.60)GOTO 30                                                 00003140
      FAIL=.TRUE.                                                         00003150
      RETURN                                                              00003160
   30 CONTINUE                                                            00003170
      P=A(L,L)-X                                                          00003180
      Q=A(L+1,L)                                                          00003190
      R=0.                                                                00003200
      LK2=L+2                                                             00003210
```

```
                      CALL QR(A,N,MAXL,P,Q,R,L,LK2,VECT)                                00003220
                      IF(ABS(A(L+2,L+1)).GT.EPS*(ABS(A(L+1,L+1))+ABS(A(L+2,L+2))))     00003230
60                   1GOTO 20                                                           00003240
                      A(L+2,L+1)=0.                                                     00003250
                      RETURN                                                            00003260
                   40 CONTINUE                                                          00003270
                   C                                                                    00003280
65                 C     ECHANGE 2*2                                                   00003290
                   C                                                                    00003300
                      M=L+2                                                             00003310
                      IF(IB2.EQ.2)M=M+1                                                 00003320
                      X=A(L+1,L+1)                                                      00003330
70                    Y=A(L,L)                                                          00003340
                      W=A(L+1,L)*A(L,L+1)                                               00003350
                      P=1.                                                              00003360
                      Q=1.                                                              00003370
                      R=1.                                                              00003380
75                    CALL QR(A,N,MAXL,P,Q,R,L,M,VECT)                                  00003390
                      IT=0                                                              00003400
                   50 IT=IT+1                                                           00003410
                      IF(IT.LE.60)GOTO 60                                               00003420
                      FAIL=.TRUE.                                                       00003430
80                    RETURN                                                            00003440
                   60 CONTINUE                                                          00003450
                      Z=A(L,L)                                                          00003460
                      R=X-Z                                                             00003470
                      S=Y-Z                                                             00003480
85                    P=(R*S-W)/A(L+1,L)+A(L,L+1)                                       00003490
                      Q=A(L+1,L+1)-Z-R-S                                                00003500
                      R=A(L+2,L+1)                                                      00003510
                      S=ABS(P)+ABS(Q)+ABS(R)                                            00003520
                      P=P/S                                                             00003530
90                    Q=Q/S                                                             00003540
                      R=R/S                                                             00003550
                      CALL QR(A,N,MAXL,P,Q,R,L,M,VECT)                                  00003560
                      IF(ABS(A(M-1,M-2)).GT.EPS*(ABS(A(M-1,M-1))+ABS(A(M-2,M-2))))     00003570
                     1  GOTO 50                                                        00003580
95                    A(M-1,M-2)=0.                                                     00003590
                      RETURN                                                            00003600
                      END                                                               00003610
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 ECHANGE

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	EPS	REAL		F.P.
0	FAIL	LOGICAL		F.P.	330	I	INTEGER		
0	IB1	INTEGER		F.P.	0	IB2	INTEGER		F.P.
333	IT	INTEGER			326	J	INTEGER		
0	L	INTEGER		F.P.	332	LK2	INTEGER		
322	L1	INTEGER			334	M	INTEGER		

```
      SUBROUTINE QR(A,N,MAXL,P,Q,R,NL,NU,VECT)                          00003620
      DIMENSION A(MAXL,MAXL),VECT(MAXL,MAXL)                            00003630
C$    ARRAYS                                                            00003640
C                                                                       00003650
C     CALCUL D UN QR IMPLICITE SUR LA MATRICE A HESSENBERG              00003660
C                                                                       00003670
      LOGICAL LAST                                                      00003680
      NL2=NL+2                                                          00003690
      DO 10 I=NL2,NU                                                    00003700
      A(I,I-2)=0.                                                       00003710
   10 CONTINUE                                                          00003720
      IF(NL2.EQ.NU)GOTO 30                                              00003730
      NL3=NL+3                                                          00003740
      DO 20 I=NL3,NU                                                    00003750
      A(I,I-3)=0.                                                       00003760
   20 CONTINUE                                                          00003770
   30 CONTINUE                                                          00003780
      NUM1=NU-1                                                         00003790
      DO 130 K=NL,NUM1                                                  00003800
      LAST=K.EQ.NUM1                                                    00003810
      IF(K.EQ.NL)GOTO 40                                                00003820
      P=A(K,K-1)                                                        00003830
      Q=A(K+1,K-1)                                                      00003840
      R=0.                                                              00003850
      IF(.NOT.LAST)R=A(K+2,K-1)                                         00003860
      X=ABS(P)+ABS(Q)+ABS(R)                                            00003870
      IF(X.EQ.0.)GOTO 130                                               00003880
      P=P/X                                                             00003890
      Q=Q/X                                                             00003900
      R=R/X                                                             00003910
   40 CONTINUE                                                          00003920
      S=SQRT(P**2+Q**2+R**2)                                            00003930
      IF(P.LT.0.)S=-S                                                   00003940
      IF(K.EQ.NL)GOTO 50                                                00003950
      A(K,K-1)=-S*X                                                     00003960
      GOTO 60                                                           00003970
   50 CONTINUE                                                          00003980
      IF(NL.NE.1)A(K,K-1)=-A(K,K-1)                                     00003990
   60 CONTINUE                                                          00004000
      P=P+S                                                             00004010
      X=P/S                                                             00004020
      Y=Q/S                                                             00004030
      Z=R/S                                                             00004040
      Q=Q/P                                                             00004050
      R=R/P                                                             00004060
      DO 80 J=K,N                                                       00004070
      P=A(K,J)+Q*A(K+1,J)                                               00004080
      IF(LAST)GOTO 70                                                   00004090
      P=P+R*A(K+2,J)                                                    00004100
      A(K+2,J)=A(K+2,J)-P*Z                                             00004110
   70 CONTINUE                                                          00004120
      A(K+1,J)=A(K+1,J)-P*Y                                             00004130
      A(K,J)=A(K,J)-P*X                                                 00004140
   80 CONTINUE                                                          00004150
C                                                                       00004160
      J=MIN0(K+3,NU)                                                    00004170
      DO 100  I=1,J                                                     00004180
```

```
                P=X*A(I,K)+Y*A(I,K+1)                                   00004190
                IF(LAST)GOTO 90                                         00004200
60              P=P+Z*A(I,K+2)                                          00004210
                A(I,K+2)=A(I,K+2)-P*R                                   00004220
             90 CONTINUE                                                00004230
                A(I,K+1)=A(I,K+1)-P*Q                                   00004240
                A(I,K)=A(I,K)-P                                         00004250
65           100 CONTINUE                                               00004260
                DO 120 I=1,N                                            00004270
                P=X*VECT(I,K)+Y*VECT(I,K+1)                             00004280
                IF(LAST)GOTO 110                                        00004290
                P=P+Z*VECT(I,K+2)                                       00004300
70              VECT(I,K+2)=VECT(I,K+2)-P*R                             00004310
             110 CONTINUE                                               00004320
                VECT(I,K+1)=VECT(I,K+1)-P*Q                             00004330
                VECT(I,K)=VECT(I,K)-P                                   00004340
             120 CONTINUE                                               00004350
75           130 CONTINUE                                               00004360
                RETURN                                                  00004370
                END                                                     00004380
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 QR

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	242	I	INTEGER		
252	J	INTEGER			245	K	INTEGER		
240	LAST	LOGICAL			0	MAXL	INTEGER		F.P.
0	N	INTEGER		F.P.	0	NL	INTEGER		F.P.
241	NL2	INTEGER			243	NL3	INTEGER		
0	NU	INTEGER		F.P.	244	NUM1	INTEGER		
0	P	REAL		F.P.	0	Q	REAL		F.P.
0	R	REAL		F.P.	247	S	REAL		
0	VECT	REAL	ARRAY	F.P.	246	X	REAL		
250	Y	REAL			251	Z	REAL		

EXTERNALS	TYPE	ARGS
SQRT	REAL	1 LIBRARY

INLINE FUNCTIONS	TYPE	ARGS					
ABS	REAL	1	INTRIN	MIN0	INTEGER	0	INTRIN

STATEMENT LABELS

0	10	0	20	32	30
75	40	112	50	116	60
140	70	0	80	165	90
0	100	205	110	0	120
211	130				

```
      SUBROUTINE RACINE(A,N,MAXL,L,E1,E2,VECT)                          00004390
      DIMENSION A(MAXL,MAXL),VECT(MAXL,MAXL)                            00004400
C$    ARRAYS                                                            00004410
      X=A(L+1,L+1)                                                      00004420
      Y=A(L,L)                                                          00004430
      W=A(L,L+1)*A(L+1,L)                                               00004440
      P=(Y-X)/2.                                                        00004450
      Q=P**2+W                                                          00004460
      IF(Q.GE.0.)GOTO 5                                                 00004470
      E1=P+X                                                            00004480
      E2=SQRT(-Q)                                                       00004490
      RETURN                                                            00004500
    5 CONTINUE                                                          00004510
      Z=SQRT(Q)                                                         00004520
      IF(P.LT.0.)GOTO 10                                                00004530
      Z=P+Z                                                             00004540
      GOTO 20                                                           00004550
   10 CONTINUE                                                          00004560
      Z=P-Z                                                             00004570
   20 CONTINUE                                                          00004580
      IF(Z.EQ.0.)GOTO 30                                                00004590
      R=-W/Z                                                            00004600
      GOTO 40                                                           00004610
   30 CONTINUE                                                          00004620
      R=0.                                                              00004630
   40 CONTINUE                                                          00004640
      IF(ABS(X+2).GE.ABS(X+R))Z=R                                       00004650
      Y=Y-X-Z                                                           00004660
      X=-Z                                                              00004670
      T=A(L,L+1)                                                        00004680
      U=A(L+1,L)                                                        00004690
      IF(ABS(Y)+ABS(U).LE.ABS(T)+ABS(X))GOTO 60                         00004700
      Q=U                                                               00004710
      P=Y                                                               00004720
      GOTO 70                                                           00004730
   60 CONTINUE                                                          00004740
      Q=X                                                               00004750
      P=T                                                               00004760
   70 CONTINUE                                                          00004770
      R=SQRT(P**2+Q**2)                                                 00004780
      IF(R.GT.0.)GOTO 80                                                00004790
      E1=A(L,L)                                                         00004800
      E2=A(L+1,L+1)                                                     00004810
      A(L+1,L)=0.                                                       00004820
      RETURN                                                            00004830
   80 CONTINUE                                                          00004840
      P=P/R                                                             00004850
      Q=Q/R                                                             00004860
      DO 90 J=L,N                                                       00004870
      Z=A(L,J)                                                          00004880
      A(L,J)=P*Z+Q*A(L+1,J)                                             00004890
      A(L+1,J)=P*A(L+1,J)-Q*Z                                           00004900
   90 CONTINUE                                                          00004910
      L1=L+1                                                            00004920
      DO 100 I=1,L1                                                     00004930
      Z=A(I,L)                                                          00004940
      A(I,L)=P*Z+Q*A(I,L+1)                                             00004950
```

```
                    A(I,L+1)=P*A(I,L+1)-Q*Z                                  00004960
                100 CONTINUE                                                 00004970
     60             DO 110 I=1,N                                             00004980
                    Z=VECT(I,L)                                              00004990
                    VECT(I,L)=P*Z+Q*VECT(I,L+1)                              00005000
                    VECT(I,L+1)=P*VECT(I,L+1)-Q*Z                            00005010
                110 CONTINUE                                                 00005020
     65             A(L+1,L)=0.                                              00005030
                    E1=A(L,L)                                                00005040
                    E2=A(L+1,L+1)                                            00005050
                    RETURN                                                   00005060
                    END                                                      00005070
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 RACINE

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	E1	REAL		F.P.
0	E2	REAL		F.P.	176	I	INTEGER		
174	J	INTEGER			0	L	INTEGER		F.P.
175	L1	INTEGER			0	MAXL	INTEGER		F.P.
0	N	INTEGER		F.P.	166	P	REAL		
167	Q	REAL			171	R	REAL		
172	T	REAL			173	U	REAL		
0	VECT	REAL	ARRAY	F.P.	165	W	REAL		
163	X	REAL			164	Y	REAL		
170	Z	REAL							

EXTERNALS	TYPE	ARGS
SQRT	REAL	1 LIBRARY

INLINE FUNCTIONS	TYPE	ARGS
ABS	REAL	1 INTRIN

STATEMENT LABELS

26	5	32	10	34	20
37	30	40	40	66	60
71	70	105	80	0	90
0	100	0	110		

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
115	90	J	49 53	4B	INSTACK
131	100	I	55 59	4B	INSTACK
145	110	I	60 64	4B	INSTACK

STATISTICS
PROGRAM LENGTH 177B 127
55000B SCM USED

II.1.6 - Résultats numériques

Nous prenons comme exemple d'application une matrice déjà utilisée dans |41| et choisie pour son mauvais conditionnement. Nous l'avons choisie volontairement de dimension 16x16 afin de mettre en défaut l'algorithme QR sur un ordinateur CDC 7600 (48 bits de mantisse) (voir Tableau I). Une dimension 13x13 nous aurait suffi sur IBM (24 bits de mantisse).

16.	15.	14.	13.	12.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
15.	15.	14.	13.	12.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	14.	14.	13.	12.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	13.	13.	12.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	12.	12.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	11.	11.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	10.	10.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	9.	9.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	8.	8.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	7.	7.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	6.	6.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	5.	5.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	4.	4.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	3.	3.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	2.	2.	1.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	1.

Tableau I

Les valeurs propres théoriques sont toutes réelles et telles que :

$$\lambda_p = 1/\lambda_{N-p+1} \qquad (2.1.28)$$

$$\text{si } |\lambda_i| > |\lambda_{i+1}| \quad \forall_i = 1,n-1 \qquad (2.1.29)$$

La construction de cette matrice ainsi que l'utilisation de l'algorithme AQRCT ont été réalisées à l'aide du programme suivant :

```
      1              DIMENSION A(20,20),VP(20,2),VECT(20,20),ITYPE(20)                  00000060
                     N=16                                                               00000070
                     IA=20                                                              00000080
                     DO3 I=1,N                                                          00000090
      5              DO 1 J=1,N                                                         00000100
                     VECT(I,J)=0.                                                       00000110
                1    A(I,J)=0.                                                          00000120
                3    VECT(I,I)=1.                                                       00000130
                     DO 2 I=1,N                                                         00000140
     10              DO 20 J=I,N                                                        00000150
                20   A(I,J)=N-J+1                                                       00000160
                     IF(I.EQ.1)GOTO 2                                                   00000170
                     A(I,I-1)=N-I+1                                                     00000180
                2    PRINT 21,(A(I,J),J=1,N)                                            00000190
     15         21   FORMAT(20F4.0)                                                     00000200
                     EPS=2.**(-47.)                                                     00000210
                     PRINT *,EPS                                                        00000215
                     CALL AQRCT(A,N,IA,1,N,EPS,VP,ITYPE,VECT)                           00000220
                     PRINT 7                                                            00000230
     20         7     FORMAT(1H1,5X,2HNO,5X,13HPARTIE REELLE,21X,*PARTIE IMAGINAIRE*)  00000240
                     DO 4 I=1,N                                                         00000250
                4    PRINT 6,VP(I,1),VP(I,2)                                            00000260
                6    FORMAT(/2X,   3X,E22.15,6X,   14X,E22.15)                          00000270
                     STOP                                                               00000280
     25              END                                                                00000290
```

CARD NR. SEVERITY DETAILS DIAGNOSIS OF PROBLEM

1 I *PROGRAM START. (INPUT,OUTPUT)* ASSUMED WHEN HEADER STATEMENT IS DEFECTIVE OR OMITTED.

SYMBOLIC REFERENCE MAP (R=1)

VARIABLES	SN	TYPE	RELOCATION					
237	A	REAL	ARRAY	236	EPS	REAL		
234	I	INTEGER		233	IA	INTEGER		
1747	ITYPE	INTEGER	ARRAY	235	J	INTEGER		
232	N	INTEGER		1127	VECT	REAL	ARRAY	
1057	VP	REAL	ARRAY					

FILE NAMES	MODE			
0	INPUT	20	OUTPUT	MIXED

EXTERNALS	TYPE	ARGS
AQRCT		9

STATEMENT LABELS

0	1	101	2		0	3	
0	4	220	6	FMT	204	7	FMT
0	20	174	21	FMT			

Les résultats numériques obtenus sont présentés dans le tableau II :

NO	PARTIE REELLE	PARTIE IMAGINAIRE
	.459467813717524E+02	0.
	.319139198874852E+02	0.
	.221414832578321E+02	0.
	.148991730381934E+02	0.
	.951384272387821E+01	0.
	.563269227412241E+01	0.
	.302378705666791E+01	0.
	.146871775236036E+01	0.
	.680865920880567E+00	0.
	.330720673638258E+00	0.
	.177246556026738E+00	0.
	.107890200238332E+00	0.
	.590481262925897E-01	.130296424624117E-01
	.590481262925897E-01	-.130296424624117E-01
	.223915171262565E-01	.813954911313858E-02
	.223915171262565E-01	-.813954911313858E-02

Tableau II

Nous remarquons immédiatement que les plus petites valeurs propres sont trouvées complexes.

Seule, la connaissance des valeurs théoriques nous permet d'affirmer que certains des résultats sont faux. Par ailleurs, nous ne pouvons pas évaluer la précision sur les valeurs propres réelles trouvées si nous possédons seulement les résultats numériques fournis par le programme.

Remarque

Si nous prenions la peine de calculer la trace de la matrice initiale et que nous comparions avec la somme des valeurs propres, nous trouverions une différence négligeable. Cette technique est donc inutilisable dans de pareils cas.

Pourquoi des valeurs propres complexes ?

Nous sommes en présence d'une matrice mal conditionnée, c'est-à-dire que les petites variations introduites par les erreurs d'arrondi ou de troncature provoquent de grandes variations au niveau des résultats (propagation des erreurs d'arrondi ou de troncature). Les matrices A_s (2.1.1) ne sont plus semblables à A, mais à une matrice (A+E) où E est une matrice erreur (ou matrice de bruit).

Le test d'arrêt (2.1.26) peut être alors décrit sous la forme :

$$|a_{i,i-1} + e_{i,i-1}| < \varepsilon.(|a_{i-1,i-1}| + |a_{i,i}|) \qquad (2.1.28)$$

où $e_{i,i-1}$ est l'erreur numérique attachée au résultat exact $a_{i,i-1}$ sur le calcul du coefficient sous-diagonal à la ligne i, au bout d'un nombre d'itérations s.

La même remarque s'applique pour le test (2.1.27).

Que se passe-t-il si (2.1.28) n'est pas vérifié pour $i = \ell$, mais que (2.1.26) le soit ? (Nous supposons que ℓ est le plus grand terme de la matrice courante).

Deux cas se présentent :

1) Le test (2.1.28) n'est pas vérifié pour $i = \ell-1$. Alors une itération inutile est effectuée.

2) Le test (2.1.28) est vérifié pour $i = \ell-1$ (ce cas de figure ne doit pas en principe arriver car la convergence s'effectue plus rapidement sur le coefficient $a_{\ell,\ell-1}$ que sur $a_{\ell-1,\ell-2}$. Il se présente neanmoins pour notre cas test). Alors un bloc 2x2 est isolé, de la forme :

$$\begin{vmatrix} \lambda_1 & C \\ r & \lambda_2 \end{vmatrix} \qquad \text{avec} \quad \begin{matrix} \lambda_1 \\ \lambda_2 \end{matrix} \quad \text{valeurs propres réelles}$$

$$\text{et } r = a_{\ell,\ell-1} + e_{\ell,\ell-1}$$

Nous cherchons alors deux valeurs propres complexes conjuguées ;

$$\lambda \pm i\,\mu \qquad \text{avec } i = \sqrt{-1}$$

$$\text{et où } \lambda = \frac{\lambda_1 + \lambda_2}{2} \quad \text{et} \quad \mu^2 = \left[\left(\frac{(\lambda_1 - \lambda_2)}{2} \right)^2 \right] + Cr,$$

soit :

$$\mu \simeq \frac{|\lambda_1 - \lambda_2|}{2}$$

Nous nous apercevons que l'erreur sur ce calcul est grande devant la valeur de $e_{\ell,\ell-1}$.

Peut-on éviter cela ?

Nous pourrions penser qu'il est suffisant d'augmenter le coefficient ε jusqu'à une valeur seuil pour laquelle le test usuel sur les coefficients nuls sous-diagonaux soit positif quel que soit le bruit numérique sur A. Mais peut-on évaluer cette valeur seuil ? De plus, en procédant ainsi, nous risquons d'augmenter considérablement les erreurs d'approximation.

Nous allons donc, dans le paragraphe suivant, chercher à optimiser ce test d'arrêt et estimer ce bruit numérique.

II.2 - TESTS D'ARRET OPTIMAUX

II.2.1 - Nécessité d'un bon test d'arrêt

Le dilemme est le suivant :

- Doit-on minimiser les erreurs d'approximations et donc prendre une valeur ε minimale ?

- Ou bien doit-on minimiser les erreurs d'arrondi ou de troncature, et donc les itérations, en choisissant une valeur pour ε relativement grande en permettant ainsi une meilleure séparation des valeurs propres réelles et complexes (réf. II.1.6) ?

Quelle que soit la valeur de ε choisie pour l'application des tests (2.1.26) et (2.1.27), cette valeur n'aura pas un caractère général. Au lieu de choisir une valeur pour ε constante, il est plus efficace que cette valeur soit fonction du bruit numérique sur le coefficient sous-diagonal testé.

Supposons l'estimation du bruit réalisée et cherchons alors un test d'arrêt efficace.

II.2.2 - Expression des tests d'arrêt optimaux

Soit x, une valeur théorique exacte, et X la valeur numérique calculée correspondante. Nous pouvons alors écrire :

$$X = x + \delta$$

où δ sera l'erreur numérique faite sur le calcul de x.

En employant la terminologie de la théorie du signal, nous écrirons encore :

$$R = S + B$$

où R est la réponse ou le résultat du calcul de x,
S est le signal ou valeur de x,
B est le bruit ou erreur numérique sur x.

Prenons maintenant comme valeur de x la valeur théorique de notre coefficient sous-diagonal de A testé, soit : $a_{\ell,\ell-1}$.

Il est clair que si S < B, une itération QR n'aura pour effet que de propager la valeur de R, c'est-à-dire surtout du bruit sur l'ensemble des autres coefficients de la matrice A ; ce serait donc une itération inutile.

Si maintenant S > B, un test d'arrêt positif va entraîner une erreur d'approximation de l'ordre du signal, donc de l'ordre de grandeur de R puisque la valeur du signal après un nombre infini d'itérations devrait être 0.

Le problème se pose alors en ce sens : à partir de quel rapport signal/bruit le signal doit-il être considéré comme négligeable ?

Si nous effectuons une itération supplémentaire, nous allons encore générer et propager du bruit et nous obtiendrons :

$$R' = S' + B'$$

mais aussi tous les coefficients seront entachés de l'erreur générée et propagée issue de R et de l'itération supplémentaire.

Nous pensons qu'a priori, l'erreur supplémentaire introduite sera du même ordre de grandeur que B et qu'elle n'excèdera pas cette valeur puisque B est issu de toutes les itérations précédentes.

Nous retiendrons donc qu'à partir de :

$$S/B \leqslant 2$$

le signal peut être considéré comme négligeable et qu'il s'agit bien d'un test optimal adapté au calcul numérique sur ordinateur.

II.2.3 - Problème de mise en oeuvre

Le problème est que la seule valeur dont nous disposons est R, S et B étant inconnus. Il faut donc pouvoir estimer S et B. Le calcul direct n'est pas possible, sinon le problème n'aurait pas de sens. La seule façon qui nous reste d'estimer S et B est donc une étude statistique. Encore faut-il créer une famille de résultats R_i indépendants.

Les travaux de M. LA PORTE et J. VIGNES |22| vont nous être d'un grand secours, la théorie des permutations-perturbations permettant cette étude statistique.

II.3 - METHODE DE PERTURBATION

II.3.1 - Principe

Nous rappelons brièvement l'idée de base des méthodes permutation-perturbation décrites dans |37| et |38|.

Tout élément $x \in \mathbb{R}$ est représenté en machine par un élément $X \in \mathbb{F}$, où $\mathbb{F}$ est l'ensemble des flottants normalisés représentables en machine. Cette valeur X sera une approximation de x.

En fait, x est écrit en virgule flottante normalisée sous la forme $x = m \times b^e$ où m est une mantisse de longueur illimitée et telle que $1/b \leqslant m < 1$, e un exposant entier relatif, et b la base.

La représentation machine de x sera de la forme $X = M * b^E$ où M sera une mantisse limitée à p digits et E un exposant entier relatif borné. Les erreurs numériques sont essentiellement dues à cette représentation limitée des nombres en machine et elles apparaissent après chaque opération élémentaire.

Une conséquence immédiate est que les règles de l'algèbre telles que l'associativité de l'addition algébrique ne sont plus satisfaites.

a) Méthode de Permutation

Considérons une méthode numérique directe définie par :

$$\text{Procédure (d,r,+,-,x,:,fonct)} \qquad (2.3.1)$$

d étant l'ensemble des données, $d \subset R$
r le résultat cherché, $r \in R$

Mise en oeuvre sur ordinateur, c'est-à-dire écrite sous une forme syntaxique équivalente à la forme algébrique, cette procédure s'écrit :

$$\text{PROCEDURE (D,R,}\oplus\text{,}\ominus\text{,*,/,FONCT)} \qquad (2.3.2)$$

avec $D \subset \mathbb{F}$ et $R \in \mathbb{F}$.

Puisque les règles de l'algèbre ne sont plus vérifiées, toutes les procédures informatiques correspondant à toutes les permutations possibles des opérateurs permutables seront chacune aussi représentatives de la procédure algébrique.

Aussi, si l'on appelle C_{op} le nombre total des combinaisons correspondant aux diverses permutations possibles des opérateurs, on peut dire qu'il y a C_{op} procédures informatiques images de la procédure algébrique.

b) Méthode de perturbation

Considérons l'une quelconque des procédures informatiques (2.3.2) image de la procédure (2.3.1) et faisons-la exécuter par la machine. Chaque opération entraîne une erreur d'affectation (par défaut ou par excès lorsque la machine travaille en arithmétique avec arrondi, par défaut seul pour une arithmétique avec troncature).

Or, si nous prenons le cas d'une arithmétique avec troncature, le résultat par excès est aussi légitime. Nous devons donc considérer que toute opération informatique admet deux résultats, l'un par défaut, l'autre par excès.

Ainsi, si k est le nombre total d'opérations élémentaires (affectations, opérations arithmétiques, fonctions) de la procédure informatique, cette dernière fournira pour R non pas un seul résultat, mais 2^k résultats tous aussi représentatifs du résultat algébrique r.

c) Méthode de permutation-perturbation

Pour obtenir l'ensemble $\mathfrak{R}$ des résultats informatiques images du résultat unique r d'un algorithme algébrique donné, il suffit d'appliquer la méthode de perturbation à chacun des algorithmes informatiques fournis par la méthode de permutation.

Le nombre d'éléments de la population $\mathfrak{R}$ ainsi obtenue est égal à :

$$\text{Card } \mathfrak{R} = 2^k C_{op} \qquad (2.3.3)$$

Il existe donc Card $\mathfrak{R}$ images informatiques du résultat unique r d'une procédure algébrique.

d) Evaluation statistique de l'erreur sur le résultat

Si R_i est un élément quelconque de $\mathfrak{R}$, le nombre C_i de chiffres décimaux significatifs du résultat peut être calculé par :

$$C_i = \text{Log}_{10} \frac{|R_i|}{\varepsilon_i} \qquad (2.3.4)$$

avec

$$\varepsilon_i = \sqrt{(R_i - \bar{R})^2 + \delta^2}$$

où ε_i est l'évaluation statistique de l'erreur commise sur R_i, $\bar{R}$ étant la moyenne des éléments de la population $\mathfrak{R}$, δ^2 étant la variance de cette population.

Maillé |25| a montré que le meilleur estimateur de R_i est donné par sa valeur moyenne $\bar{R}$. C_i peut alors être estimé par :

$$C = \mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} \qquad (2.3.5)$$

Toujours dans |25|, il est montré que les résultats R_i peuvent être considérés comme une variable aléatoire Gaussienne de moyenne $\bar{R}$ et d'écart-type δ.

Nous écrirons :

$$R_i = \bar{R} + (R_i - \bar{R}) = S + B_i \qquad (2.3.6)$$

avec B_i variable aléatoire Gaussienne centrée d'écart type δ, B_i est un bruit numérique lié au signal S du résultat théorique exact r.

L'intervalle de confiance à p % du résultat r exact défini par la loi de Student est donné par :

$$P[r \in [\bar{R} - t_p \frac{\delta}{\sqrt{N}}, \bar{R} + t_p \frac{\delta}{\sqrt{N}}]] = 1 - \frac{P}{100} \qquad (2.3.7)$$

où t_p est la valeur de la distribution de Student pour (N-1) degrés de liberté.

Pour N = 3 et p = 5, t_p = 4.3 (Table de Student). L'intervalle de confiance est alors défini par $r \in I_c$, où

$$I_c = [\bar{R} - 2.48\ \delta, \bar{R} + 2{,}48\ \delta] \qquad (2.3.8)$$

Le nombre de chiffres décimaux significatifs exact C_e est égal à :

$$C_e = \mathrm{Log}_{10} \frac{|\bar{R}|}{e} \quad \text{avec } \frac{e}{2} = 2.48 \times \delta \qquad (2.3.9)$$

soit :

$$C_e = \mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} - 0.69624\ldots = C - 0.69624 \qquad (2.3.10)$$

Nous ne chercherons pas une grande précision sur le nombre de chiffres décimaux significatifs. L'estimation de C est plus un moyen de contrôle de l'erreur numérique qu'un calcul exact. La précision obtenue sur C avec trois éléments de $\mathfrak{R}$ nous paraît satisfaisante. Prendre plus d'éléments pénaliserait trop les performances de nos algorithmes.

Enfin, pour des raisons de complexité, nous n'appliquerons pas de permutations sur l'algorithme QR, nous nous limiterons aux perturbations.

II.3.2 - Mise en oeuvre de la méthode de perturbation pour l'application d'un test d'arrêt optimal

De (2.3.6) nous constatons que le signal S peut être estimé par la moyenne $\bar{R}$ des résultats R_i obtenus.

Le niveau du bruit B sera alors estimé par δ, écart-type de la variable aléatoire R_i. Nous identifierons donc :

$$\mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} \text{ à } \mathrm{Log}_{10} \left|\frac{S}{B}\right|$$

Tester $S/B \leqslant 2$ revient alors à tester :

$$\boxed{\mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} \lesssim 0.3} \qquad (2.3.11)$$

Il est évident que le choix de l'estimateur de S/B est relativement imprécis, néanmoins l'estimation de l'ordre de grandeur reste correcte, et cela nous suffit.

Nous pourrions augmenter la population des résultats R_i afin d'avoir un test plus précis. Mais nous arrivons vite à une pénalisation en calcul et en place mémoire pour un gain de précision faible.

Pour mettre en place nos tests d'arrêt en utilisant la méthode de perturbation, il est nécessaire de garder en mémoire trois matrices semblables à la matrice A et d'effectuer simultanément les calculs sur ces trois matrices en affectant aléatoirement un 0 ou un 1 au dernier bit de la mantisse de tout résultat fourni par chaque opération élémentaire $[\oplus, \ominus, *, /]$. (Pour une machine CDC 7600 ; arithmétique avec troncature).

La comparaison des résultats à chaque itération sur les coefficients sous-diagonaux nous permettra alors de séparer le signal ($\bar{R}$) du bruit (δ).

Le test (2.1.26) devient alors :

$$\mathrm{Log}_{10} \frac{|\bar{R}(a_{\ell,\ell-1})|}{\delta(a_{\ell,\ell-1})} \leqslant 0.3 \qquad (2.3.12)$$

et le test (2.1.27) deviendra :

$$\mathrm{Log}_{10} \frac{|\bar{R}(a_{m,m-1}(|q_m| + |r_m|))|}{\delta(a_{m,m-1}(|q_m| + |r_m|))} \leqslant 0.3 \qquad (2.3.13)$$

Il ne faut voir dans l'utilisation du log décimal qu'un moyen de pouvoir calculer en même temps le nombre de chiffres significatifs.

Nous aurons besoin pour cela de quelques programmes utilitaires que nous définirons dans le paragraphe suivant.

Remarque importante

L'algorithme QR pour les matrices réelles quelconques nécessite une mise en forme préalable de Hessenberg supérieure. En toute rigueur, on ne peut négliger les erreurs introduites par cet algorithme. D'autres erreurs peuvent être introduites avant cette mise en forme (calculs divers, résultats statistiques, etc.).

Nous préférons donc dissocier l'algorithme QR de celui de la mise en forme en matrice de Hessenberg supérieure, et nous laissons la possibilité d'effectuer les itérations QR à partir de trois matrices représentatives des résultats numériques possibles.

Nous donnerons plus loin un moyen simple d'obtenir ces trois matrices représentatives des résultats numériques pour l'algorithme de mise en forme de Hessenberg supérieure.

II.3.3 - Programmes utilitaires

Une perturbation aléatoire du dernier digit de la mantisse d'un nombre représenté en machine est très dépendante de la machine. En effet, la représentation machine d'un nombre réel varie d'un constructeur à l'autre. Avec un CDC 7600, nous utilisons les fonctions XV et HASARD déjà utilisées dans |37|.

La fonction HASARD permet d'effectuer un tirage aléatoire de 0 ou de 1, XV ajoute ou pas 1 au dernier bit de la mantisse du nombre à perturber. Pour lier ces deux fonctions, nous utilisons la fonction PER(X) perturbant le contenu de X, qui est une version du programme PEPER toujours défini dans |37|.

Pour appliquer ces perturbations de façon qu'il n'y ait ni redondance ni oubli, nous avons décomposé l'algorithme QR en une série d'opérations élémentaires, chaque opération étant effectuée lors de l'appel de la fonction PER.

Exemple : pour Z = Y + Y, nous écrirons :

$$Z = PER(X+Y) \qquad (2.3.14)$$

Nous avons été amené à considérer la fonction SQRT comme une opération élémentaire et la fonction ABS comme une non-opération.

Pour l'application des tests (2.3.12) et (2.3.13), nous utiliserons le sous-programme WXTEST testant le nombre de chiffres significatifs sur un résultat numérique à partir d'un échantillon de résultats.

```
      FUNCTION XV(X,S)
      PRE(Z)=OR(0000400000000000000B,AND(ABS(Z),7777000000000000000B))
      DER(Z)=OR(0000000000000000001B,AND(ABS(Z),7777000000000000000B))
      IF(X.EQ.0.)GOTO 1
      XV=X+SIGN(DER(X),S)
      IF(PRE(X).GT.PRE(XV))XV=XV-SIGN(DER(XV),S)
      RETURN
1     XV=SIGN(PRE(0.),S)
      RETURN
      END
```

```
      FUNCTION PER (PR)
      B=2.-1.E-13
      IPILE=HASARD(0.,B)
       IF(IPILE.EQ.0)GOTO 1
        IPR=PR
      PRS=IPR
      IF(PRS.EQ.PR)GOTO 1
      PER =XV(PR,SIGN(FLOAT(IPILE),PR))
      GOTO 2
1     PER =PR
2     RETURN
      END
```

```
      FUNCTION HASARD(DEBUT,FIN)
      EQUIVALENCE (X,I)
      COMMON/HAS001/N,X,XR(3),C,CR(3)
      DATA N,X,XR,C,CR/0,4*1001.,4*0./
      C=C+1.
      V=MOD(I,10000000)
      W=MOD(I,10000001)
      V=V+(W-5.E6)/1.E7+0.5
      X=C/(V+1.)
      HASARD=(DEBUT*(1.E7-V)+FIN*V)/1.E7
      RETURN
      END
```

Appel

CALL WXTEST(R,N,C,RM),RETURNS(ETIQUETTE)

Arguments d'entrée

R Tableau monodimensionné, de dimension au moins égale à N, contenant différents résultats numériques représentant une même valeur théorique.

N Nombre de valeurs numériques rentrées.

Arguments de sortie

C Nombre de chiffres significatifs des résultats numériques, en valeur réelle.

RM Valeur moyenne.

Retour

Retour à l'ETIQUETTE indiquée si $C \leqslant 0.3$.

II.3.4 - Notice d'utilisation du programme CHSIQR

Objet

Calcul des valeurs propres par ordre de module décroissant par la méthode QR d'une matrice Hessenberg supérieure, avec conservation des transformations, test d'arrêt optimal et calcul du nombre de chiffres significatifs sur les valeurs propres obtenues.

Appel

CALL CHSIQR(A,N,IA,INF,ISUP,VP,CHSIVP,VECT,ITYPE,IARG)

Arguments d'entrée

A Bloc réel tridimensionné dont les deux premières dimensions sont IA et la troisième égale à 3, contenant les éléments de la matrice Hessenberg supérieure avec $A(I,J,1) = a_{ij}$; $i,j = 1,\ldots,N$.
Selon IARG, 3 estimations peuvent être entrées; dans ce cas A(I,J,1), A(I,J,2), A(I,J,3) seront les estimations de a_{ij}. A sera détruite.

N entier, ordre de la matrice (a_{ij}).

IA 1ère dimension des blocs A,VP,CHSIVP,VECT,ITYPE,
2ème dimension de A et VECT.

INF,ISUP Les calculs ne seront effectués que sur le bloc diagonal (a_{ij}) avec i,j = INF, INF+1,...ISUP.
Nous avons $1 \leqslant INF < ISUP \leqslant N$.

VECT Bloc réel tridimensionné dont les deux premières dimensions sont IA et la troisième égale à 3, contenant les éléments de la matrice de transformation issue de la mise en forme Hessenberg supérieure. Par défaut, mettre la matrice unité. Selon IARG, 3 estimations peuvent être données.

IARG = 0 : A(I,J,2) et A(I,J,3) seront initialisées automatiquement à A(I,J,1) ; idem pour VECT.
= 1 : A(I,J,2) et A(I,J,3) seront gardées ; idem pour VECT.

Arguments de sortie

A Contient les 3 matrices perturbées mises sous la forme quasi-triangulaire supérieure.

VP Bloc réel bidimensionné dont la première dimension est IA et la deuxième 2.
VP(I,1) contient la partie réelle de la valeur propre λ_i,
VP(I,2) contient sa partie imaginaire.

CHSIVP Bloc réel de mêmes dimensions que VP.
CHSIVP (I,1) contient le nombre de chiffres significatifs de VP(I,1),
CHSIVP (I,2) pour VP(I,2).

ITYPE Bloc entier de dimensions au moins égales à N contenant -1, 0, 1 ou 2 :
-1: valeur propre non trouvée
0 : valeur propre réelle
1 : valeur propre complexe à partie imaginaire positive
2 : valeur propre complexe à partie imaginaire négative

VECT Contient les 3 matrices des transformations dans le même ordre que pour A.

L'organigramme de CHSIVP est identique à celui de AQRCT.

II.3.5 - Programme FORTRAN CHSIQR (1ère version)

```
      SUBROUTINE CHSIGR(A,N,MAXL,INF,ISUP,VP,CHSIVP,VECT,ITYPE,IARG)
      DIMENSION A(MAXL,MAXL,3),VP(MAXL,2),CHSIVP(MAXL,2)
     %,VECT(MAXL,MAXL,3),ITYPE(MAXL)
      DIMENSION E1(3),E2(3),PP(3),QQ(3),RR(3)
      COMMON/CHSIRQ/T(3),X(3),Y(3),W(3),WX(3)
      LOGICAL COMP
      LOGICAL FAIL
C
C
C     INITIALISATION
C
      NS=3
      IF(INF.EQ.1)GOTO 1111
      INF1=INF-1
      DO 1101 K=1,INF1
      VP(K,1)=A(K,K,1)
1101  VP(K,2)=0.
1111  IF(ISUP.EQ.N)GOTO 2222
      ISUP1=ISUP+1
      DO 1103 K=ISUP1,N
      VP(K,1)=A(K,K,1)
1103  VP(K,2)=0.
2222  CONTINUE
      DO 2233 I=1,N
      CHSIVP(I,1)=15.
      CHSIVP(I,2)=15.
      IF(IARG.EQ.1)GOTO 2233
      DO 2223 J=1,N
      DO 2223 IS=2,NS
      VECT(I,J,IS)=VECT(I,J,1)
2223  A(I,J,IS)=A(I,J,1)
2233  CONTINUE
      DO 10 I=INF,ISUP
10    ITYPE(I)=-1
      DO 2224 IS=1,NS
2224  T(IS)=0.
C
C     RECHERCHE ET MISE EN ORDRE DES VALEURS PROPRES
C
      NU=ISUP
100   IF(NU.LT.INF)GOTO 500
      IT=0
C
C     ELEMENTS SOUS-DIAGONAUX NEGLIGEABLES
C
110   L=NU
120   CONTINUE
      IF(L.EQ.INF)GOTO 130
      DO 121 IS=1,NS
121   WX(IS)=A(L,L-1,IS)
      CALL WXTEST(WX,NS,C,RM),RETURNS(130)
      L=L-1
      GOTO 120
C
C     EST CE 1 BLOC 1*1 OU 2*2 QUI SE DETACHE
C
130   CONTINUE
```

```
      DO 131 IS=1,NS
  131 X(IS)=A(NU,NU,IS)
60    IF(L.EQ.NU)GOTO 300
      DO 132 IS=1,NS
      Y(IS)=A(NU-1,NU-1,IS)
  132 W(IS)=PER(A(NU,NU-1,IS)*A(NU-1,NU,IS))
      IF(L.EQ.NU-1)GOTO 200
65    IF(IT.EQ.60)GOTO 500
C
C     SI PLUS DE 20 ITERATIONS ALORS TRANSLATION DE L'ORIGINE SPECIALE
C
      IF(IT.NE.20.AND.IT.NE.40)GOTO 150
70    DO 151 IS=1,NS
      T(IS)=PER(T(IS)+X(IS))
      DO 140 I=INF,NU
  140 A(I,I,IS)=PER(A(I,I,IS)-X(IS))
      S=PER(ABS(A(NU,NU-1,IS))+ABS(A(NU-1,NU-2,IS)))
75    X(IS)=PER(0.75*S)
      Y(IS)=X(IS)
      S=PER(S*S)
  151 W(IS)=PER(-0.4375*S)
  150 CONTINUE
80    IT=IT+1
C
C     RECHERCHE DE DEUX ELEMENTS SOUS-DIAGONAUX NEGLIGEABLES
C
      NL=NU-2
85    160 CONTINUE
      DO 161 IS=1,NS
      Z=A(NL,NL,IS)
      R=PER(X(IS)-Z)
      S=PER(Y(IS)-Z)
90    P=PER(R*S)
      P=PER(P-W(IS))
      P=PER(P/A(NL+1,NL,IS))
      P=PER(P+A(NL,NL+1,IS))
      Q=PER(A(NL+1,NL+1,IS)-Z)
95    Q=PER(Q-R)
      Q=PER(Q-S)
      R=A(NL+2,NL+1,IS)
      S=PER(ABS(P)+ABS(Q))
      S=PER(S+ABS(R))
100   P=PER(P/S)
      Q=PER(Q/S)
      R=PER(R/S)
       PP(IS)=P
      QQ(IS)=Q
105   RR(IS)=R
  161 CONTINUE
      IF(NL.EQ.L)GOTO 170
      DO 162 IS=1,NS
      WX(IS)=PER(ABS(Q)+ABS(R))
110   162 WX(IS)=PER(A(NL,NL-1,IS)*WX(IS))
      CALL WXTEST(WX,NS,C,RM),RETURNS(170)
      NL=NL-1
      GOTO 160
  170 CONTINUE
```

```
      C
      C     ITERATION QR ENTRE NL ET NU
      C
      C
      C     BLOC 2*2
      C
            DO 171 IS=1,NS
            P=PP(IS)
            Q=QQ(IS)
            R=RR(IS)
171         CALL PERQR(A,N,MAXL,P,Q,R,NL,NU,VECT,IS)
            GOTO 110
      C
      C     UN BLOC 2*2
      C
200         CONTINUE
            DO 201 IS=1,NS
            IF(NU.NE.INF+1)A(NU-1,NU-2,IS)=0.
            A(NU,NU,IS)=A(NU,NU,IS)+T(IS)
201         A(NU-1,NU-1,IS)=A(NU-1,NU-1,IS)+T(IS)
            ITYPE(NU)=0
            ITYPE(NU-1)=0
            MU=NU
210         CONTINUE
            NL=MU-1
      C
      C     S'AGIT IL DE DEUX VALEURS PROPRES REELLES
      C
            DO 209 IS=1,NS
209         CALL RACINE(A,N,MAXL,NL,E1,E2,VECT,IS)
      C
      C     SI OUI ON VA STOCKER CES DEUX VALEURS
      C
            IF(A(MU,MU-1,1).EQ.0.)GOTO 310
            IF(MU.EQ.ISUP)GOTO 400
            IF(MU.EQ.ISUP-1)GOTO 220
            IF(A(MU+2,MU+1,1).EQ.0.)GOTO 220
      C
      C     LE PROCHAIN BLOC EST 2*2.
      C
            DO 211 IS=1,NS
            WX(IS)=A(MU-1,MU-1,IS)*A(MU,MU,IS)-A(MU-1,MU,IS)
           %*A(MU,MU-1,IS)-A(MU+1,MU+1,IS)*A(MU+2,MU+2,IS)
           %+A(MU+1,MU+2,IS)*A(MU+2,MU+1,IS)
211         CONTINUE
            CALL WXTEST(WX,NS,C,RM),RETURNS(444)
444         IF(RM.GE.0)GOTO 400
            CALL ECHANGE(A,N,MAXL,NL,2,2,FAIL,VECT,NS)
            IF(.NOT.FAIL)GOTO 215
            ITYPE(NL)=-1
            ITYPE(NL+1)=-1
            ITYPE(NL+2)=-1
            ITYPE(NL+3)=-1
            GOTO 500
215         CONTINUE
            MU=MU+2
            GOTO 210
```

```
      220   CONTINUE
      C
      C     BLOC 1*1
175   C
            DO 221 IS=1,NS
            WX(IS)=A(MU-1,MU-1,IS)*A(MU,MU,IS)-A(MU-1,MU,IS)
           %*A(MU,MU-1,IS)-A(MU+1,MU+1,IS)**2
      221   CONTINUE
180         CALL WXTEST(WX,NS,C,RM),RETURNS(445)
      445   IF(RM.GE.0)GOTO 400
            CALL ECHANGE(A,N,MAXL,NL,2,1,FAIL,VECT,NS)
            IF(.NOT.FAIL)GOTO 225
            ITYPE(NL)=-1
185         ITYPE(NL+1)=-1
            ITYPE(NL+2)=-1
            GOTO 500
      225   CONTINUE
            MU=MU+1
190         GOTO 210
      C
      C     UNE VALEUR PROPRE A ETE TROUVEE
      C
      300   NL=0
195         DO 311 IS=1,NS
            A(NU,NU,IS)=A(NU,NU,IS)+T(IS)
      311   IF(NU.NE.INF)A(NU,NU-1,IS)=0.
            ITYPE(NU)=0
            MU=NU
200   310   CONTINUE
      320   CONTINUE
            IF(MU.EQ.ISUP)GOTO 350
            IF(MU.EQ.ISUP-1)GOTO 330
            IF(A(MU+2,MU+1,1).EQ.0.)GOTO 330
205         DO 321 IS=1,NS
            WX(IS)=A(MU,MU,IS)**2-A(MU+1,MU+1,IS)*A(MU+2,MU+2,IS)
           %+A(MU+1,MU+2,IS)*A(MU+2,MU+1,IS)
        321  CONTINUE
            CALL WXTEST(WX,NS,C,RM),RETURNS(446 )
210   446   IF(RM.GE.0)GOTO 400
            CALL ECHANGE(A,N,MAXL,MU,1,2,FAIL,VECT,NS)
            IF(.NOT.FAIL) GOTO 325
            ITYPE(MU)=-1
            ITYPE(MU+1)=-1
215         ITYPE(MU+2)=-1
            GOTO 500
      325   CONTINUE
            MU=MU+2
            GOTO 320
220   C
      C     BLOC 1*1
      C
      330   CONTINUE
            DO 331 IS=1,NS
225         WX(IS)=ABS(A(MU,MU,IS))-ABS(A(MU+1,MU+1,IS))
      331   CONTINUE
            CALL WXTEST(WX,NS,C,RM),RETURNS(447)
      447    IF(RM.GE.0.)GOTO 350
```

```
                CALL ECHANGE(A,N,MAXL,MU,1,1,FAIL,VECT,NS)
230             MU=MU+1
                GOTO 320
              350 CONTINUE
                MU=NL
                NL=0
235             IF(MU.NE.0)GOTO 310
              400 CONTINUE
                NU=L-1
                GOTO 100
            C
240         C     TOUTES LES VALEURS PROPRES ONT ETE TROUVEES ET ORDONNEES
            C
            500 IF(NU.LT.INF)GOTO 503
                DO 504 IS=1,NS
                DO 504 I=1,NU
245         504 A(I,I,IS)=A(I,I,IS)+T(IS)
            503 CONTINUE
                NU=ISUP
            510 CONTINUE
                IF(ITYPE(NU).NE.-1)GOTO 515
250             NU=NU-1
                GOTO 540
            515 CONTINUE
                IF(NU.EQ.INF)GOTO 520
                IF(A(NU,NU-1,1).EQ.0.)GOTO 520
255         C
            C     BLOC 2*2
            C
                NUM1=NU-1
                DO 511 IS=1,NS
260         511 CALL RACINE(A,N,MAXL,NUM1,E1,E2,VECT,IS)
                IF(A(NU,NU-1,1).EQ.0.)GOTO 520
                CALL WXTEST(E1,NS,CHSIVP(NU,1),VP(NU,1)),RETURNS(512)
            512 CALL WXTEST(E2,NS,CHSIVP(NU-1,2),VP(NU-1,2)),RETURNS(513)
            513 CHSIVP(NU-1,1)=CHSIVP(NU,1)
265             CHSIVP(NU,2)=CHSIVP(NU-1,2)
                VP(NU-1,1)=VP(NU,1)
                VP(NU,2)=VP(NU-1,2)
                ITYPE(NU-1)=1
                ITYPE(NU)=2
270             NU=NU-2
                GOTO 530
            C
            C      UNE RACINE SIMPLE
            C
275          520 CONTINUE
                DO 521 IS=1,NS
            521 E1(IS)=A(NU,NU,IS)
                CALL WXTEST(E1,NS,CHSIVP(NU,1),VP(NU,1)),RETURNS(522)
            522 VP(NU,2)=0.
280             NU=NU-1
              530 CONTINUE
            540 CONTINUE
                IF(NU.GE.INF)GOTO 510
                RETURN
285             END
```

```
      SUBROUTINE ECHANGE(A,N,MAXL,L,IB1,IB2,FAIL,VECT,NS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3)
      COMMON/CHSIRQ/T(3),X(3),Y(3),W(3),WX(3)
C
      LOGICAL FAIL
      FAIL=.FALSE.
      IF(IB1.EQ.2)GOTO 40
      IF(IB2.EQ.2)GOTO 10
C
C     ECHANGE DE DEUX BLOCS 1*1
C
      L1=L+1
      DO 1 IS=1,NS
      Q=PER(A(L+1,L+1,IS)-A(L,L,IS))
      P=A(L,L+1,IS)
      R=AMAX1(P,Q)
      IF(R.EQ.0.)RETURN
      P=PER(P/R)
      Q=PER(Q/R)
      R=PROSCAL(P,Q,P,Q)
      R=PER(SQRT(R))
      P=PER(P/R)
      Q=PER(Q/R)
      DO 3 J=L,N
      S=PROSCAL(P,Q,A(L,J,IS),A(L+1,J,IS))
      A(L+1,J,IS)=PROSCAL(P,-Q,A(L+1,J,IS),A(L,J,IS))
3     A(L,J,IS)=S
      DO 5 I=1,L1
      S=PROSCAL(P,Q,A(I,L,IS),A(I,L+1,IS))
      A(I,L+1,IS)=PROSCAL(P,-Q,A(I,L+1,IS),A(I,L,IS))
5     A(I,L,IS)=S
      DO 7 I=1,N
      S=PROSCAL(P,Q,VECT(I,L,IS),VECT(I,L+1,IS))
      VECT(I,L+1,IS)=PROSCAL(P,-Q,VECT(I,L+1,IS),VECT(I,L,IS))
7     VECT(I,L,IS)=S
1     A(L+1,L,IS)=0.
      RETURN
C
C     ECHANGE 1*1 2*2
C
10    CONTINUE
      LK2=L+2
      DO 2 IS=1,NS
      X(IS)=A(L,L,IS)
      P=1.
      Q=1.
      R=1.
2     CALL PERQR(A,N,MAXL,P,Q,R,L,LK2,VECT,IS)
      IT=0
20    IT=IT+1
      IF(IT.LE.60)GOTO 30
      FAIL=.TRUE.
      RETURN
30    CONTINUE
      LK2=L+2
      DO 4 IS=1,NS
      P=PER(A(L,L,IS)-X(IS))
```

```
                Q=A(L+1,L,IS)
                R=0.
 60             CALL PERQR(A,N,MAXL,P,Q,R,L,LK2,VECT,IS)
       4        WX(IS)=A(L+2,L+1,IS)
                CALL WXTEST(WX,NS,C,RM),RETURNS(80)
                GOTO 20
       80       CONTINUE
 65             DO 6 IS=1,NS
       6        A(L+2,L+1,IS)=0.
                RETURN
       C
       C        ECHANGE 2*2
 70    C
       40       M=L+2
                IF(IB2.EQ.2)M=M+1
                DO 8 IS=1,NS
                X(IS)=A(L+1,L+1,IS)
 75             Y(IS)=A(L,L,IS)
                W(IS)=PER(A(L+1,L,IS)*A(L,L+1,IS))
                P=1.
                Q=1.
                R=1.
 80     8       CALL PERQR(A,N,MAXL,P,Q,R,L,M,VECT,IS)
                IT=0
       50       IT=IT+1
                IF(IT.LE.60) GOTO 60
                FAIL=.TRUE.
 85             RETURN
       60       CONTINUE
                DO 9 IS=1,NS
                Z=A(L,L,IS)
                R=PER(X(IS)-Z)
 90             S=PER(Y(IS)-Z)
                P=PER(R*S)
                P=PER(P-W(IS))
                P=PER(P/A(L+1,L,IS))
                P=PER(P+A(L,L+1,IS))
 95             Q=PER(A(L+1,L+1,IS)-Z)
                Q=PER(Q-R)
                Q=PER(Q-S)
                R=A(L+2,L+1,IS)
                S=PER(ABS(P)+ABS(Q))
100             S=PER(ABS(R)+S)
                P=PER(P/S)
                Q=PER(Q/S)
                R=PER(R/S)
                 CALL PERQR(A,N,MAXL,P,Q,R,L,M,VECT,IS)
105      9      WX(IS)=A(M-1,M-2,IS)
                CALL WXTEST(WX,NS,C,RM),RETURNS(70)
                GOTO 50
       70       CONTINUE
                DO 11 IS=1,NS
110      11     A(M-1,M-2,IS)=0.
                RETURN
                END
```

```
      1              FUNCTION PROSCAL(U1,U2,V1,V2)
                     U=PER(U1*V1)
                     V=PER(U2*V2)
                     PROSCAL=PER(U+V)
      5              RETURN
                     END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 PROSCAL

VARIABLES	SN	TYPE	RELOCATION				
27	PROSCAL	REAL		30	U	REAL	
0	U1	REAL	F.P.	0	U2	REAL	F.P.
31	V	REAL		0	V1	REAL	F.P.
0	V2	REAL	F.P.				

EXTERNALS	TYPE	ARGS
PER	REAL	1

STATISTICS
PROGRAM LENGTH 32B 26
55000B SCM USED

```
      SUBROUTINE PER QR(A,N,MAXL,P,Q,R,NL,NU,VECT,IS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3)
      LOGICAL LAST
C
C
C     CALCUL D'UNE ITERATION QR SUR A HESSENBERG
C
      NL2=NL+2
      DO 10 I=NL2,NU
10    A(I,I-2,IS)=0.
      IF(NL2.EQ.NU)GOTO 30
      NL3=NL+3
      DO 20 I=NL3,NU
20    A(I,I-3,IS)=0.
30    CONTINUE
      NUM1=NU-1
      DO 130 K=NL,NUM1
      LAST=K.EQ.NUM1
      IF(K.EQ.NL)GOTO 40
      P=A(K,K-1,IS)
      Q=A(K+1,K-1,IS)
      R=0.
      IF(.NOT.LAST)R=A(K+2,K-1,IS)
      X=PER(ABS(P)+ABS(Q))
      X=PER(X+ABS(R))
      IF(X.EQ.0.)GOTO 130
      P=PER(P/X)
      Q=PER(Q/X)
      R=PER(R/X)
40    S=PROSCAL(P,Q,P,Q)
      V=PER(R**2)
         S=PER(S+V)
      S=PER(SQRT(S))
      IF(P.LT.0.)S=-S
      IF(K.EQ.NL)GOTO 50
      A(K,K-1,IS)=PER(-S*X)
      GOTO 60
50    CONTINUE
      IF(NL.NE.1)A(K,K-1,IS)=-A(K,K-1,IS)
60    CONTINUE
      P= PER(P+S)
      X= PER(P/S)
      Y= PER(Q/S)
      Z= PER(R/S)
      Q= PER(Q/P)
      R= PER(R/P)
C
C     MULTIPLICATION A DROITE,PUIS AGAUCHE
C
      DO 80 J=K,N
      P=PER(Q*A(K+1,J,IS))
      P=PER(P+A(K,J,IS))
      IF(LAST)GOTO 70
      V=PER(R*A(K+2,J,IS))
      P=PER(P+V)
      V=PER(P*Z)
      A(K+2,J,IS)=PER(A(K+2,J,IS)-V)
```

```
                70    V=PER(P*Y)
                      A(K+1,J,IS)=PER(A(K+1,J,IS)-V)
60                    V=PER(P*X)
                80    A(K,J,IS)=PER(A(K,J,IS)-V)
                C
                      J=MIN0(K+3,NU)
                C
65                    DO 100 I=1,J
                      P=PER(X*A(I,K,IS))
                      V=PER(Y*A(I,K+1,IS))
                      P=PER(P+V)
                      IF(LAST)GOTO 90
70                    V=PER(Z*A(I,K+2,IS))
                      P=PER(P+V)
                      V=PER(P*R)
                      A(I,K+2,IS)=PER(A(I,K+2,IS)-V)
                90    V=PER(P*Q)
75                    A(I,K+1,IS)=PER(A(I,K+1,IS)-V)
                100   A(I,K,IS)=PER(A(I,K,IS)-P)
                C
                C     CONSERVATION DES TRANSFORMATIONS
                C
80                    DO 120 I=1,N
                      P=PER(X*VECT(I,K,IS))
                      V=PER(Y*VECT(I,K+1,IS))
                      P=PER(P+V)
                      IF(LAST)GOTO 110
85                    V=PER(Z*VECT(I,K+2,IS))
                      P=PER(P+V)
                      V=PER(P*R)
                      VECT(I,K+2,IS)=PER(VECT(I,K+1,IS)-V)
                110    V=PER(P*Q)
90                    VECT(I,K+1,IS)=PER(VECT(I,K+1,IS)-V)
                      VECT(I,K  ,IS)=PER(VECT(I,K  ,IS)-P)
                120   CONTINUE
                130    CONTINUE
                      RETURN
95                    END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 PERQR

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	621	I	INTEGER		
0	IS	INTEGER		F.P.	632	J	INTEGER		
624	K	INTEGER			617	LAST	LOGICAL		
0	MAXL	INTEGER		F.P.	0	N	INTEGER	F.P.	
0	NL	INTEGER		F.P.	620	NL2	INTEGER		
622	NL3	INTEGER			0	NU	INTEGER	F.P.	
623	NUM1	INTEGER			0	P	REAL	F.P.	
0	Q	REAL		F.P.	0	R	REAL	F.P.	

```
      SUBROUTINE RACINE(A,N,MAXL,L,E1,E2,VECT,IS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3),E1(3),E2(3)
C
      X=A(L+1,L+1,IS)
      Y=A(L,L,IS)
      W=PER(A(L,L+1,IS)*A(L+1,L,IS))
      P=PER(Y-X)
      P=P/2.
      Q=PER(P**2)
      Q=PER(Q+W)
      IF(Q.GE.0.)GOTO 5
      E1(IS)=PER(P+X)
      E2(IS)=PER(SQRT(-Q))
      RETURN
C
    5 CONTINUE
      Z=PER(SQRT(Q))
      IF(P.LT.0.)GOTO 10
      Z=PER(P+Z)
      GOTO 20
   10 Z=PER(P-Z)
   20 CONTINUE
      IF(Z.EQ.0.)GOTO 30
      R=PER(-W/Z)
      GOTO 40
   30 R=0.
   40 CONTINUE
      IF(ABS(PER(X+Z)).GE.ABS(PER(X+R)))Z=R
      Y=PER(Y-X)
      Y=PER(Y-Z)
      X=-Z
      T=A(L,L+1,IS)
      U=A(L,L+1,IS)
      IF(PER(ABS(Y)+ABS(U)).LE.PER(ABS(T)+ABS(X)))GOTO 60
      Q=U
      P=Y
      GOTO 70
   60 Q=X
      P=T
   70 R=PROSCAL(P,Q,P,Q)
      R=PER(SQRT(R))
      IF(R.GT.0.)GOTO 80
      E1(IS)=A(L,L,IS)
      E2(IS)=A(L+1,L+1,IS)
      A(L+1,L,IS)=0.
      RETURN
C
   80 P=PER(P/R)
      Q=PER(Q/R)
      DO 90 J=L,N
      Z=A(L,J,IS)
      A(L,J,IS)=PROSCAL(P,Q,Z,A(L+1,J,IS))
   90 A(L+1,J,IS)=PROSCAL(P,-Q,A(L+1,J,IS),Z)
      L1=L+1
      DO 100 I=1,L1
      Z=A(I,L,IS)
      A(I,L,IS)=PROSCAL(P,Q,Z,A(I,L+1,IS))
```

```
      100   A(I,L+1,IS)=PROSCAL(P,-Q,A(I,L+1,IS),Z)
            DO 110 I=1,N
60          Z=VECT(I,L,IS)
            VECT(I,L)=PROSCAL(P,Q,Z,VECT(I,L+1,IS))
      110   VECT(I,L+1)=PROSCAL(P,-Q,VECT(I,L+1,IS),Z)
            A(L+1,L,IS)=0.
            E1(IS)=A(L,L,IS)
65          E2(IS)=A(L+1,L+1,IS)
            RETURN
            END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 RACINE

VARIABLES	SN	TYPE		RELOCATION						
0	A	REAL	ARRAY	F.P.	0	E1	REAL	ARRAY	F.P.	
0	E2	REAL	ARRAY	F.P.	446	I	INTEGER			
0	IS	INTEGER		F.P.	444	J	INTEGER			
0	L	INTEGER		F.P.	445	L1	INTEGER			
0	MAXL	INTEGER		F.P.	0	N	INTEGER		F.P.	
436	P	REAL			437	Q	REAL			
441	R	REAL			442	T	REAL			
443	U	REAL			0	VECT	REAL	ARRAY	F.P.	
435	W	REAL			433	X	REAL			
434	Y	REAL			440	Z	REAL			

EXTERNALS	TYPE	ARGS				
PER	REAL	1		PROSCAL	REAL	4
SQRT	REAL	1	LIBRARY			

INLINE FUNCTIONS	TYPE	ARGS	
ABS	REAL	1	INTRIN

STATEMENT LABELS

62	5	73	10	100	20		
105	30	106	40	163	60		
166	70	211	80	0	90		
0	100	0	110				

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
226	90	J	50 53	32B	EXT REFS
265	100	I	55 58	33B	EXT REFS
323	110	I	59 62	30B	EXT REFS

STATISTICS
PROGRAM LENGTH 512B 330
55000B SCM USED

III.3.6 - Résultats numériques

Reprenons l'exemple-test de II.1.6 accompagné du programme d'appel suivant :

```
      PROGRAM   TEST1(INPUT,OUTPUT,TAPE60=INPUT,TAPE61=OUTPUT)
      DIMENSION A(20,20,3),VP(20,2),CHSIVP(20,2),VECT(20,20,3)
     %,ITYPE(20)
C
      N=16
      PRINT 8
    8 FORMAT(1H1)
      IA=20
      DO 1 I=1,N
      DO 2 J=1,I
    2 A(I,J,1)=0.
      DO 3 J=I,N
    3 A(I,J,1)=N-J+1
      DO 5 J=1,N
    5 VECT(I,J,1)=0.
      VECT(I,I,1)=1.
      IF(I.EQ.1)GOTO 1
      A(I,I-1,1)=N-I+1
    1 PRINT 9,(A(I,J,1),J=1,N)
    9 FORMAT(20F4.0)
      PRINT 8
       IARG=0
      CALL CHSIQR(A,N,IA,1,N,VP,CHSIVP,VECT,ITYPE,IARG)
      PRINT 7
    7  FORMAT(1H1,5X,2HNO,5X,13HPARTIE REELLE,7X,11HCHIFF.SIGN.,10X,
     %  17HPARTIE IMAGINAIRE,3X,11HCHIFF.SIGN.)
      DO 4 I=1,N
      N1=CHSIVP(I,1)
      N2=CHSIVP(I,2)
    4 PRINT 6,I,VP(I,1),N1,VP(I,2),N2
    6 FORMAT(/2X,I5,3X,E21.14,7X,I2,14X,E21.14,7X,I2)
      END
```

Nous avons alors les résultats numériques suivants :

Tableau III

NO	PARTIE REELLE	CHIFF.SIGN.	PARTIE IMAGINAIRE	CHIFF.SIGN.
1	.45946781371779E+02	13	0.	15
2	.31913919887503E+02	13	0.	15
3	.22141483257848E+02	13	0.	15
4	.14899173038202E+02	13	0.	15
5	.95138427238847E+01	13	0.	15
6	.56326922741262E+01	13	0.	15
7	.30237870481410E+01	8	0.	15
8	.14687177618093E+01	8	0.	15
9	.68086594262359E+00	7	0.	15
10	.33070761613793E+00	4	0.	15
11	.17782416039951E+00	2	0.	15
12	.99354224580058E-01	1	0.	15
13	.82814989816868E-01	0	0.	15
14	.66492689418924E-01	0	0.	15
15	.68814851091683E-02	0	0.	15
16	.14661528615941E-01	0	0.	15

Nous constatons que :

- Les fausses valeurs propres complexes ont disparu. La séparation entre signal et bruit sur les coefficients sous-diagonaux a donc bien été faite.

- Nous n'avons pu éviter la perte de précision. En effet, le mauvais conditionnement de la matrice reste et nous n'avons pas ici diminué le nombre global d'itérations ; il faudrait pour cela passer à des matrices d'ordre bien supérieur.

- Nous sommes mieux renseignés sur la précision sur les valeurs propres obtenues. Nous pourrions vérifier ces estimations sur un passage double précision effectué dans II.4.4. Nous verrons alors que ces résultats sont acceptables.

Quels sont les problèmes rencontrés ?

- Non simplicité de mise en oeuvre.

- Rattachement aux calculs effectués en amont.

- Accroissement important du coût : appel aux fonctions à chaque opération.

Nous allons considérer que la méthode employée est la méthode de référence et nous allons chercher une méthode économique adaptée.

II.4 - METHODE DE MULTIPLICATION

II.4.1 - Principe

Soit D, un ensemble de données numériques, F, un algorithme, F(D), le résultat numérique de l'algorithme F appliqué aux données D.

Supposons qu'il existe une fonction g telle que si k est un facteur multiplicatif, la valeur théorique de F(k.D) soit égale à g(k).F(D). Si nous posons e = (F(k*D)/g(k)-F(D)), si e $\neq$ 0, et enfin si nous pouvons considérer que les erreurs numériques introduites par les opérations k*D, g(k) et la division de F(k*D) par g(k) sont négligeables devant l'erreur due au calcul de F(D) ou de F(k.D), alors e est représentatif d'une erreur numérique sur F(D).

Il est important de savoir si e est de même ordre de grandeur que l'erreur numérique sur F(D).

Nous pouvons dire que si k n'est pas une puissance de 2, la représentation en virgule flottante d'un nombre (a) quelconque est très différente du nombre (k.a) et qu'une opération élémentaire ($\oplus$, $\ominus$,*,/) d'un nombre (a) avec un nombre (b) générera une erreur d'arrondi ou de troncature différente de celle provenant d'une opération entre (k.a) et (k.b) si cette erreur n'est pas nulle.

Soit alors a un nombre réel, ℓ(a) le nombre de bits utiles de sa mantisse.

Nous définissons le nombre de bits utiles comme le nombre de bits comptés à partir du 1er bit à 1 trouvé jusqu'au dernier bit à 1 en parcourant la mantisse dans un sens ou dans l'autre.

Exemple

Soit a dont la mantisse est de représentation binaire : 1 0 1 1 0 0 0... Nous aurons ℓ(a) = 4.

Proposition

Soit m le maximum de bits utilisables dans une mantisse (dépendant-machine).

Soit a et k tels que $\ell(a) + \ell(k) \leqslant m$.

Alors, pour c = a * k $\quad \ell(c) \leqslant \ell(a) + \ell(k)$ (2.4.1)

Démonstration

Soit n(a) le nombre entier correspondant à la représentation binaire contenue dans les ℓ(a) bits utiles de la mantisse de a.

Si $\ell(a) > 1$, alors :

$$2^{\ell(a)-1} < n(a) < 2^{\ell(a)} \qquad (2.4.2)$$

Avant arrondi ou troncature, nous avons : $n(c) = n(a)\ n(k)$, d'où :

$$2^{\ell(a)+\ell(k)-2} < n(c) < 2^{\ell(a)+\ell(k)} \leqslant 2^{m} \qquad (2.4.3)$$

Comme avant arrondi ou troncature il y a une normalisation effectuée, c'est-à-dire calage à gauche du premier bit trouvé à 1 et que, ainsi, la longueur de bits utiles de c n'excède pas m, il n'y a pas arrondi ou troncature

Si $\ell(a) + \ell(k) > m$, nous montrons facilement que le nombre de bits perdus de c, noté p(c), par dépassement de capacité de la mantisse est tel que :

$$0 \leqslant p(c) \leqslant \ell(a) + \ell(k) - m \qquad (2.4.4)$$

Remarque

La précision sur c reste de m bits.

Néanmoins, nous allons minimiser la perte de précision p(c).

Choix d'un facteur k multiplicatif

Il nous faut minimiser p(c) (réf. 2.4.4) pour a quelconque. Il faut donc choisir k tel que $\ell(k)$ soit minimum (voir également 2.4.4) et qu'il modifie suffisamment la configuration de la mantisse de a*k par rapport à a.

Fonctionnant en représentation binaire, les puissances de 2 ne modifieront que la partie exposant.

Si nous choissons un exposant nul, le seul représentant de $\ell(k) = 2$, est 3. Pour $\ell(k) = 3$, nous aurons 5. et 7.

Choisissons $k_1 = 1.$, $k_2 = 3.$, $k_3 = 5.$

Alors k_i*D, i = 1,3 a une perte maximale d'information de trois bits par rapport à D, et possède le même nombre de chiffres significatifs.

Nous allons alors comparer les résultats obtenus par cette méthode à ceux obtenus par la méthode de perturbation et montrer que les solutions :

$$S_i = (F(k_i * D)/g(k_i)) \qquad (2.4.5)$$

ont la même dispersion que les résultats $R_i \in \mathcal{R}$, où $\mathcal{R}$ est l'ensemble des résultats numériques possibles de F(D).

Pour ce faire, nous allons tester notre méthode sur l'algorithme QR, et vérifier que les résultats numériques obtenus sont comparables à ceux obtenus par la méthode de perturbation.

II.4.2 - Mise en oeuvre de la méthode de multiplication pour l'application d'un test d'arrêt optimal

Les résultats numériques à tester sont les coefficients sous-diagonaux $a_{\ell,\ell-1}$, q_m, r_m et les valeurs propres finales λ_i.

Que deviennent ces valeurs lorsque la matrice initiale A est remplacée par la matrice (k*A) ? Pour $a_{\ell,\ell-1}$ et λ_i, les valeurs sont remplacées par $k*a_{\ell,\ell-1}$ et $k*\lambda_i$. En effet, les matrices Q_i sont des matrices unitaires, d'où :

$$Q_s(k*A_s)\ Q_s^T = k*Q_s\ A_s\ Q_s^T = k*A_{s+1}$$

si Q_s est inchangée. C'est le cas : il suffit de se reporter à (2.1.14) et à (2.1.23) et (2.1.24) pour observer que les matrices P_2 sont théoriquement identiques, car calculées à partir de u_r et v_r vecteurs normés à 1.

Si p_m, q_m, r_m sont remplacés par k^2*p_m, k^2*q_m, k^2*r_m, comme ils sont ensuite normés, leurs valeurs théoriques sont en fait inchangées.

L'expression de (F(k*D)/g(k)) de (2.4.5) se ramène donc à :

$$(F(k*D)/k) \qquad (2.4.6)$$

pour l'étude des valeurs numériques de $a_{\ell,\ell-1}$ et $a_{\ell,\ell-1}*(|q_m| + |r_m|)$ utilisées dans les tests 1 et 2 définis en (2.3.12) et (2.3.13).

En résumé, si nous reprenons le programme CHSIQR utilisant la méthode de perturbation, nous allons faire les seules modifications suivantes :

1) Nos 3 matrices initiales ne sont plus identiques à la matrice A ; elles sont respectivement égales à k_1*A, k_2*A, k_3*A et nous prendrons $k_1 = 1.$, $k_2 = 3.$, $k_3 = 5.$

2) Nous n'effectuons plus de perturbation.

3) Avant chaque appel au sous-programme WXTEST testant les résultats numériques issus de (k_1*A), (k_2*A), (k_3*A), ces résultats seront respectivement divisés par k_1, k_2 et k_3.

Nous n'avons pas jugé utile de changer le nom du programme, la notice d'utilisation et l'organigramme restant identiques.

Par contre, afin de généraliser l'emploi de WXTEST, nous avons légèrement modifié son appel, et nous préférons redonner sa notice d'utilisation, un emploi courant de cette version étant fait par la suite.

Appel

```
CALL WXTEST(R,N,C,RM,RETOUR)

COMMON/WXTEST/CMIN,CMAX
```

Arguments d'entrée

R Tableau monodimensionné, de dimension au moins égale à N, contenant différents résultats numériques représentant une même valeur théorique.

N Nombre de valeurs numériques rentrées.

CMIN Nombre de chiffres décimaux significatifs (en valeur réelle) en deça duquel les résultats numériques sont jugés non significatifs.

CMAX Nombre maximal de chiffres décimaux utilisés sur la machine. Exemple : CDC 7600, simple précision, CMAX = 15. (par excès).

Arguments de sortie

C Nombre de chiffres décimaux significatifs des résultats numériques, en valeur réelle.

RM Résultat moyen

RETOUR Variable logique
RETOUR=.FALSE. pour C $\geqslant$ CMIN
RETOUR=.TRUE. pour C < CMIN

```
      SUBROUTINE WXTEST(R,N,C,RM,RETOUR)                                0000000
      DIMENSION R(N)                                                    0000000
      COMMON/WXTST/CMIN,CMAX                                            0000000
      LOGICAL RETOUR                                                    0000000
C                                                                       0000000
      RETOUR=.FALSE.                                                    0000000
      RM=0.                                                             0000000
      DO 1 I=1,N                                                        0000000
    1 RM=RM+R(I)                                                        0000000
      RM=RM/FLOAT(N)                                                    0000000
      VAR=0.                                                            0000000
      DO 2 I=1,N                                                        0000000
      DELTA=R(I)-RM                                                     0000000
    2 VAR=VAR+DELTA*DELTA                                               0000000
      VAR=VAR/FLOAT(N-1)                                                0000000
      ER=SQRT(VAR)                                                      0000000
      IF(ER.EQ.0.)                      GOTO 4                          0000000
      G=ABS(RM/ER)                                                      0000000
      IF(G.EQ.0.)                       GOTO 3                          0000000
      C=ALOG10(G)                                                       0000000
                                        GOTO 5                          0000000
    3 C=0.                                                              0000000
                                        GOTO 5                          0000000
    4 C=CMAX                                                            0000000
C                                                                       0000000
C     SI AUCUN CHIFFRE SIGNIFICATIF RETOUR EN M                         0000000
C                                                                       0000000
    5 IF(C.LT.CMIN)                     RETOUR=.TRUE.                   0000000
      RETURN                                                            0000000
      END                                                               0000000
```


SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 WXTEST

VARIABLES	SN	TYPE		RELOCATION					
0	C	REAL		F.P.	1	CMAX	REAL		WXTST
0	CMIN	REAL		WXTST	50	DELTA	REAL		
51	ER	REAL			52	G	REAL		
46	I	INTEGER			0	N	INTEGER		F.P.
0	R	REAL	ARRAY	F.P.	0	RETOUR	LOGICAL		F.P.
0	RM	REAL		F.P.	47	VAR	REAL		

EXTERNALS	TYPE	ARGS				
ALOG10	REAL	1	LIBRARY	SQRT	REAL	1 LIBRARY

INLINE FUNCTIONS	TYPE	ARGS				
ABS	REAL	1	INTRIN	FLOAT	REAL	1 INTRIN

STATEMENT LABELS

0	1	0	2	35	3
37	4	41	5		

II.4.3 - Programme FORTRAN CHSIQR (2ème version)

```
      SUBROUTINE CHSIQR(A,N,MAXL,INF,ISUP,VP,CHSIVP,VECT,ITYPE,IARG)
      DIMENSION A(MAXL,MAXL,3),VP(MAXL,2),CHSIVP(MAXL,2)
     %,VECT(MAXL,MAXL,3),ITYPE(MAXL)
      DIMENSION E1(3),E2(3),PP(3),QQ(3),RR(3)
      COMMON/WXTST/CMIN,CMAX
      COMMON/CHSIRQ/IMUL(3),T(3),X(3),Y(3),W(3),WX(3)
      REAL IMUL
      LOGICAL RETOUR
      LOGICAL FAIL
        DATA IMUL/1.,5.,3./
C
C
C     INITIALISATION
C
      NS=3
      CMIN=0.3
      CMAX=15.
      IF(INF.EQ.1)GOTO 1111
      INF1=INF-1
      DO 1101 K=1,INF1
      VP(K,1)=A(K,K,1)
1101  VP(K,2)=0.
1111  IF(ISUP.EQ.N)GOTO 2222
      ISUP1=ISUP+1
      DO 1103 K=ISUP1,N
      VP(K,1)=A(K,K,1)
1103  VP(K,2)=0.
2222  CONTINUE
      DO 2233 I=1,N
      CHSIVP(I,1)=15.
      CHSIVP(I,2)=15.
      IF(IARG.EQ.1)GOTO 2233
      DO 2223 J=1,N
      DO 2223 IS=2,NS
      VECT(I,J,IS)=VECT(I,J,1)
2223  CONTINUE
2233  CONTINUE
      DO 10 I=INF,ISUP
10    ITYPE(I)=-1
      DO 2224 IS=1,NS
      DO 2225 J=1,N
      DO 2225 I=1,N
2225  A(I,J,IS)=A(I,J,1)*IMUL(IS)
2224  T(IS)=0.
C
C     RECHERCHE ET MISE EN ORDRE DES VALEURS PROPRES
C
      NU=ISUP
100   IF(NU.LT.INF)GOTO 500
      IT=0
C
C     ELEMENTS SOUS-DIAGONAUX NEGLIGEABLES
C
110   L=NU
120   CONTINUE
      IF(L.EQ.INF)GOTO 130
      DO 121 IS=1,NS
```

```
      121   WX(IS)=A(L,L-1,IS)/IMUL(IS)
            CALL WXTEST(WX,NS,C,RM,RETOUR)
      60    IF(RETOUR)GOTO 130
            L=L-1
            GOTO 120
      C
      C     EST CE 1 BLOC 1*1 OU 2*2 QUI SE DETACHE
      65    C
      130   CONTINUE
            DO 131 IS=1,NS
      131   X(IS)=A(NU,NU,IS)
            IF(L.EQ.NU)GOTO 300
      70    DO 132 IS=1,NS
            Y(IS)=A(NU-1,NU-1,IS)
      132   W(IS)=(A(NU,NU-1,IS)*A(NU-1,NU,IS))
            IF(L.EQ.NU-1)GOTO 200
            IF(IT.EQ.60)GOTO 500
      75    C
      C     SI PLUS DE 20 ITERATIONS ALORS TRANSLATION DE L'ORIGINE SPECIALE
      C
            IF(IT.NE.20.AND.IT.NE.40)GOTO 150
            DO 151 IS=1,NS
      80    T(IS)=(T(IS)+X(IS))
            DO 140 I=INF,NU
      140   A(I,I,IS)=(A(I,I,IS)-X(IS))
            S=(ABS(A(NU,NU-1,IS))+ABS(A(NU-1,NU-2,IS)))
            X(IS)=(0.75*S)
      85    Y(IS)=X(IS)
            S=(S*S)
      151   W(IS)=(-0.4375*S)
      150   CONTINUE
            IT=IT+1
      90    C
      C     RECHERCHE DE DEUX ELEMENTS SOUS-DIAGONAUX NEGLIGEABLES
      C
            NL=NU-2
      160   CONTINUE
      95    DO 161 IS=1,NS
            Z=A(NL,NL,IS)
            R=(X(IS)-Z)
            S=(Y(IS)-Z)
            P=(R*S)
      100   P=(P-W(IS))
            P=(P/A(NL+1,NL,IS))
            P=(P+A(NL,NL+1,IS))
            Q=(A(NL+1,NL+1,IS)-Z)
            Q=(Q-R)
      105   Q=(Q-S)
            R=A(NL+2,NL+1,IS)
            S=(ABS(P)+ABS(Q))
            S=(S+ABS(R))
            P=(P/S)
      110   Q=(Q/S)
            R=(R/S)
             PP(IS)=P
            QQ(IS)=Q
            RR(IS)=R
```

```
115          161   CONTINUE
                   IF(NL.EQ.L)GOTO 170
                   DO 162 IS=1,NS
                   WX(IS)=(ABS(Q)+ABS(R))
             162   WX(IS)=(A(NL,NL-1,IS)*WX(IS))/IMUL(IS)
120                CALL WXTEST(WX,NS,C,RM,RETOUR)
                   IF(RETOUR)GOTO 170
                   NL=NL-1
                   GOTO 160
             170   CONTINUE
125          C
             C     ITERATION QR ENTRE NL ET NU
             C
             C
             C     BLOC 2*2
130          C
                   DO 171 IS=1,NS
                   P=PP(IS)
                   Q=QQ(IS)
                   R=RR(IS)
135          171   CALL QR(A,N,MAXL,P,Q,R,NL,NU,VECT,IS)
                   IF(IARG.NE.987654)GOTO 110
                   PRINT 3,IT,(A(NU,NU,IS)/IMUL(IS),IS=1,NS)
             3     FORMAT(/,1X,11HITERATION =,I4,2X,6HA(I,I),3(2X,E21.14))
                   GOTO 110
140          C
             C     UN BLOC 2*2
             C
             200   CONTINUE
                   DO 201 IS=1,NS
145                IF(NU.NE.INF+1)A(NU-1,NU-2,IS)=0.
                   A(NU,NU,IS)=A(NU,NU,IS)+T(IS)
             201   A(NU-1,NU-1,IS)=A(NU-1,NU-1,IS)+T(IS)
                   ITYPE(NU)=0
                   ITYPE(NU-1)=0
150                MU=NU
             210   CONTINUE
                   NL=MU-1
             C
             C     S'AGIT IL DE DEUX VALEURS PROPRES REELLES
155          C
                   DO 209 IS=1,NS
             209   CALL RACINE(A,N,MAXL,NL,E1,E2,VECT,IS)
             C
             C     SI OUI ON VA STOCKER CES DEUX VALEURS
160          C
                   IF(A(MU,MU-1,1).EQ.0.)GOTO 310
                   IF(MU.EQ.ISUP)GOTO 400
                   IF(MU.EQ.ISUP-1)GOTO 220
                   IF(A(MU+2,MU+1,1).EQ.0.)GOTO 220
165          C
             C     LE PROCHAIN BLOC EST 2*2,ON ECHANGE QU'EN CAS DE CERTITUDE
             C
                   DO 211 IS=1,NS
                   WX(IS)=(A(MU-1,MU-1,IS)*A(MU,MU,IS)-A(MU-1,MU,IS)
170               %*A(MU,MU-1,IS)-A(MU+1,MU+1,IS)*A(MU+2,MU+2,IS)
                  %+A(MU+1,MU+2,IS)*A(MU+2,MU+1,IS))/IMUL(IS)
```

```
      211   CONTINUE
            CALL WXTEST(WX,NS,C,RM,RETOUR)
            IF(RM.GE.0)GOTO 400
175         CALL ECHANGE(A,N,MAXL,NL,2,2,FAIL,VECT,NS)
            IF(.NOT.FAIL)GOTO 215
            ITYPE(NL)=-1
            ITYPE(NL+1)=-1
            ITYPE(NL+2)=-1
180         ITYPE(NL+3)=-1
            GOTO 500
      215   CONTINUE
            MU=MU+2
            GOTO 210
185   220   CONTINUE
      C
      C     BLOC 1*1
      C
            DO 221 IS=1,NS
190         WX(IS)=(A(MU-1,MU-1,IS)*A(MU,MU,IS)-A(MU-1,MU,IS)
           %*A(MU,MU-1,IS)-A(MU+1,MU+1,IS)**2)/IMUL(IS)
      221   CONTINUE
            CALL WXTEST(WX,NS,C,RM,RETOUR)
            IF(RM.GE.0.)GOTO 400
195         CALL ECHANGE(A,N,MAXL,NL,2,1,FAIL,VECT,NS)
            IF(.NOT.FAIL)GOTO 225
            ITYPE(NL)=-1
            ITYPE(NL+1)=-1
            ITYPE(NL+2)=-1
200         GOTO 500
      225   CONTINUE
            MU=MU+1
            GOTO 210
      C
205   C     UNE VALEUR PROPRE A ETE TROUVEE
      C
      300   NL=0
            DO 311 IS=1,NS
            A(NU,NU,IS)=A(NU,NU,IS)+T(IS)
210   311   IF(NU.NE.INF)A(NU,NU-1,IS)=0.
            ITYPE(NU)=0
            MU=NU
      310   CONTINUE
      320   CONTINUE
215         IF(MU.EQ.ISUP)GOTO 350
            IF(MU.EQ.ISUP-1)GOTO 330
            IF(A(MU+2,MU+1,1).EQ.0.)GOTO 330
            DO 321 IS=1,NS
            WX(IS)=(A(MU,MU,IS)**2-A(MU+1,MU+1,IS)*A(MU+2,MU+2,IS)
220        %+A(MU+1,MU+2,IS)*A(MU+2,MU+1,IS))/IMUL(IS)
        321 CONTINUE
            CALL WXTEST(WX,NS,C,RM,RETOUR)
            IF(RM.GE.0.)GOTO 400
            CALL ECHANGE(A,N,MAXL,MU,1,2,FAIL,VECT,NS)
225         IF(.NOT.FAIL) GOTO 325
            ITYPE(MU)=-1
            ITYPE(MU+1)=-1
            ITYPE(MU+2)=-1
```

```
                GOTO 500
230         325 CONTINUE
                MU=MU+2
                GOTO 320
            C
            C   BLOC 1*1
235         C
            330 CONTINUE
                DO 331 IS=1,NS
                WX(IS)=(ABS(A(MU,MU,IS))-ABS(A(MU+1,MU+1,IS)))/IMUL(IS)
            331 CONTINUE
240             CALL WXTEST(WX,NS,C,RM,RETOUR)
                IF(RM.GE.0.)GOTO 350
                CALL ECHANGE(A,N,MAXL,MU,1,1,FAIL,VECT,NS)
                MU=MU+1
                GOTO 320
245           350 CONTINUE
                MU=NL
                NL=0
                IF(MU.NE.0)GOTO 310
              400 CONTINUE
250             NU=L-1
                GOTO 100
            C
            C   TOUTES LES VALEURS PROPRES ONT ETE TROUVEES ET ORDONNEES
            C
255         500 IF(NU.LT.INF)GOTO 503
                DO 504 IS=1,NS
                DO 504 I=1,NU
            504 A(I,I,IS)=A(I,I,IS)+T(IS)
            503 CONTINUE
260             NU=ISUP
            510 CONTINUE
                IF(ITYPE(NU).NE.-1)GOTO 515
                NU=NU-1
                GOTO 540
265         515 CONTINUE
                IF(NU.EQ.INF)GOTO 520
                IF(A(NU,NU-1,1).EQ.0.)GOTO 520
            C
            C   BLOC 2*2
270         C
                NUM1=NU-1
                DO 511 IS=1,NS
            511 CALL RACINE(A,N,MAXL,NUM1,E1,E2,VECT,IS)
                IF(A(NU,NU-1,1).EQ.0.)GOTO 520
275             CALL WXTEST(E1,NS,CHSIVP(NU,1),VP(NU,1),RETOUR)
                CALL WXTEST(E2,NS,CHSIVP(NU-1,2),VP(NU-1,2),RETOUR)
                CHSIVP(NU-1,1)=CHSIVP(NU,1)
                CHSIVP(NU,2)=CHSIVP(NU-1,2)
                VP(NU-1,1)=VP(NU,1)
280             ITYPE(NU-1)=1
                VP(NU,2)=-VP(NU-1,2)
                ITYPE(NU)=2
                NU=NU-2
                GOTO 530
285         C
```

```
                C      UNE RACINE SIMPLE
                C
                 520  CONTINUE
                      DO 521 IS=1,NS
     290         521  E1(IS)=A(NU,NU,IS)/IMUL(IS)
                      CALL WXTEST(E1,NS,CHSIVP(NU,1),VP(NU,1),RETOUR)
                      VP(NU,2)=0.
                      NU=NU-1
                   530 CONTINUE
     295          540  CONTINUE
                      IF(NU.GE.INF)GOTO 510
                      DO 9999 IS=1,NS
                      DO 9999 J=1,N
                      DO 9999 I=1,N
     300         9999  A(I,J,IS)=A(I,J,IS)/IMUL(IS)
                      RETURN
                      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3. CHSIQR

VARIABLES	SN	TYPE	RELOCATION						
0	A	REAL	ARRAY	F.P.	1300	C	REAL		
0	CHSIVP	REAL	ARRAY	F.P.	1	CMAX	REAL		WXTST
0	CMIN	REAL		WXTST	1312	E1	REAL	ARRAY	
1315	E2	REAL	ARRAY		1265	FAIL	LOGICAL		
1272	I	INTEGER			0	IARG	INTEGER		F.P.
0	IMUL	REAL	ARRAY	CHSIRQ	0	INF	INTEGER		F.P.
1267	INF1	INTEGER			1274	IS	INTEGER		
0	ISUP	INTEGER		F.P.	1271	ISUP1	INTEGER		
1276	IT	INTEGER			0	ITYPE	INTEGER	ARRAY	F.P.
1273	J	INTEGER			1270	K	INTEGER		
1277	L	INTEGER			0	MAXL	INTEGER		F.P.
1310	MU	INTEGER			0	N	INTEGER		F.P.
1303	NL	INTEGER			1266	NS	INTEGER		
1275	NU	INTEGER			1311	NUM1	INTEGER		
1306	P	REAL			1320	PP	REAL	ARRAY	
1307	Q	REAL			1323	QQ	REAL	ARRAY	
1305	R	REAL			1264	RETOUR	LOGICAL		
1301	RM	REAL			1326	RR	REAL	ARRAY	
1302	S	REAL			3	T	REAL	ARRAY	CHSIRQ
0	VECT	REAL	ARRAY	F.P.	0	VP	REAL	ARRAY	F.P.
14	W	REAL	ARRAY	CHSIRQ	17	WX	REAL	ARRAY	CHSIRQ
6	X	REAL	ARRAY	CHSIRQ	11	Y	REAL	ARRAY	CHSIRQ
1304	Z	REAL							

FILE NAMES	MODE
OUTPUT	FMT

```
      SUBROUTINE ECHANGE(A,N,MAXL,L,IB1,IB2,FAIL,VECT,NS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3)
      COMMON/CHSIRQ/IMUL(3),T(3),X(3),Y(3),W(3),WX(3)
      REAL IMUL
C
      LOGICAL FAIL,RETOUR
      FAIL=.FALSE.
      PRINT *,L,A(L,L,1),IB1,IB2
      IF(IB1.EQ.2)GOTO 40
      IF(IB2.EQ.2)GOTO 10
C
C     ECHANGE DE DEUX BLOCS 1*1
C
      L1=L+1
      DO 1 IS=1,NS
      Q=(A(L+1,L+1,IS)-A(L,L,IS))
      P=A(L,L+1,IS)
      R=AMAX1(P,Q)
      IF(R.EQ.0.)RETURN
      P=(P/R)
      Q=(Q/R)
      R=P*P+Q*Q
      R=(SQRT(R))
      P=(P/R)
      Q=(Q/R)
      DO 3 J=L,N
      S=P*A(L,J,IS)+Q*A(L+1,J,IS)
      A(L+1,J,IS)=P*A(L+1,J,IS)-Q*A(L,J,IS)
    3 A(L,J,IS)=S
      DO 5 I=1,L1
      S=P*A(I,L,IS)+Q*A(I,L+1,IS)
      A(I,L+1,IS)=P*A(I,L+1,IS)-Q*A(I,L,IS)
    5 A(I,L,IS)=S
      DO 7 I=1,N
      S=P*VECT(I,L,IS)+Q*VECT(I,L+1,IS)
      VECT(I,L+1,IS)=P*VECT(I,L+1,IS)-Q*VECT(I,L,IS)
    7 VECT(I,L,IS)=S
    1 A(L+1,L,IS)=0.
      RETURN
C
C     ECHANGE 1*1 2*2
C
   10 CONTINUE
      LK2=L+2
      DO 2 IS=1,NS
      X(IS)=A(L,L,IS)
      P=1.
      Q=1.
      R=1.
    2 CALL QR(A,N,MAXL,P,Q,R,L,LK2,VECT,IS)
      IT=0
   20 IT=IT+1
      IF(IT.LE.60)GOTO 30
      FAIL=.TRUE.
      RETURN
   30 CONTINUE
      LK2=L+2
```

```
                DO 4 IS=1,NS
                P=(A(L,L,IS)-X(IS))
 60             Q=A(L+1,L,IS)
                R=0.
                CALL QR(A,N,MAXL,P,Q,R,L,LK2,VECT,IS)
          4     WX(IS)=A(L+2,L+1,IS)/IMUL(IS)
                CALL WXTEST(WX,NS,C,RM,RETOUR)
 65             IF(.NOT.RETOUR)GOTO 20
          80    CONTINUE
                DO 6 IS=1,NS
          6     A(L+2,L+1,IS)=0.
                RETURN
 70       C
          C     ECHANGE 2*2
          C
          40    M=L+2
                IF(IB2.EQ.2)M=M+1
 75             DO 8 IS=1,NS
                X(IS)=A(L+1,L+1,IS)
                Y(IS)=A(L,L,IS)
                W(IS)=(A(L+1,L,IS)*A(L,L+1,IS))
                P=1.
 80             Q=1.
                R=1.
          8     CALL QR(A,N,MAXL,P,Q,R,L,M,VECT,IS)
                IT=0
          50    IT=IT+1.
 85             IF(IT.LE.60) GOTO 60
                FAIL=.TRUE.
                RETURN
          60    CONTINUE
                DO 9 IS=1,NS
 90             Z=A(L,L,IS)
                R=(X(IS)-Z)
                S=(Y(IS)-Z)
                P=(R*S)
                P=(P-W(IS))
 95             P=(P/A(L+1,L,IS))
                P=(P+A(L,L+1,IS))
                Q=(A(L+1,L+1,IS)-Z)
                Q=(Q-R)
                Q=(Q-S)
100             R=A(L+2,L+1,IS)
                S=(ABS(P)+ABS(Q))
                S=(ABS(R)+S)
                P=(P/S)
                Q=(Q/S)
105             R=(R/S)
                 CALL QR(A,N,MAXL,P,Q,R,L,M,VECT,IS)
          9     WX(IS)=A(M-1,M-2,IS)/IMUL(IS)
                CALL WXTEST(WX,NS,C,RM,RETOUR)
                IF(.NOT.RETOUR)GOTO 50
110       70    CONTINUE
                DO 11 IS=1,NS
          11    A(M-1,M-2,IS)=0.
                RETURN
                END
```

```
      1              FUNCTION PROSCAL(U1,U2,V1,V2)
                     U=(U1*V1)
                     V=(U2*V2)
                     PROSCAL=(U+V)
      5              RETURN
                     END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS

4	PROSCAL

VARIABLES	SN	TYPE	RELOCATION					
13	PROSCAL	REAL		14	U	REAL		
0	U1	REAL	F.P.	0	U2	REAL	F.P.	
15	V	REAL		0	V1	REAL	F.P.	
0	V2	REAL	F.P.					

STATISTICS

PROGRAM LENGTH	16B	14
55000B SCM USED		

```
      SUBROUTINE  QR(A,N,MAXL,P,Q,R,NL,NU,VECT,IS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3)
      LOGICAL LAST
C
C
C     CALCUL D'UNE ITERATION QR SUR A HESSENBERG
C
      NL2=NL+2
      DO 10 I=NL2,NU
10    A(I,I-2,IS)=0.
      IF(NL2.EQ.NU)GOTO 30
      NL3=NL+3
      DO 20 I=NL3,NU
20    A(I,I-3,IS)=0.
30    CONTINUE
      NUM1=NU-1
      DO 130 K=NL,NUM1
      LAST=K.EQ.NUM1
      IF(K.EQ.NL)GOTO 40
      P=A(K,K-1,IS)
      Q=A(K+1,K-1,IS)
      R=0.
      IF(.NOT.LAST)R=A(K+2,K-1,IS)
      X=(ABS(P)+ABS(Q))
      X=(X+ABS(R))
      IF(X.EQ.0.)GOTO 130
      P=(P/X)
      Q=(Q/X)
      R=(R/X)
40    S=P*P+Q*Q
      V=(R**2)
         S=(S+V)
      S=(SQRT(S))
      IF(P.LT.0.)S=-S
      IF(K.EQ.NL)GOTO 50
      A(K,K-1,IS)=(-S*X)
      GOTO 60
50    CONTINUE
      IF(NL.NE.1)A(K,K-1,IS)=-A(K,K-1,IS)
60    CONTINUE
      P= (P+S)
      X= (P/S)
      Y= (Q/S)
      Z= (R/S)
      Q= (Q/P)
      R= (R/P)
C
C     MULTIPLICATION A DROITE,PUIS AGAUCHE
C
      DO 80 J=K,N
      P=(Q*A(K+1,J,IS))
      P=(P+A(K,J,IS))
      IF(LAST)GOTO 70
      V=(R*A(K+2,J,IS))
      P=(P+V)
      V=(P*Z)
      A(K+2,J,IS)=(A(K+2,J,IS)-V)
```

```
      70    V=(P*Y)
            A(K+1,J,IS)=(A(K+1,J,IS)-V)
60          V=(P*X)
      80    A(K,J,IS)=(A(K,J,IS)-V)
      C
            J=MIN0(K+3,NU)
      C
65          DO 100 I=1,J
            P=(X*A(I,K,IS))
            V=(Y*A(I,K+1,IS))
            P=(P+V)
            IF(LAST)GOTO 90
70          V=(Z*A(I,K+2,IS))
            P=(P+V)
            V=(P*R)
            A(I,K+2,IS)=(A(I,K+2,IS)-V)
      90    V=(P*Q)
75          A(I,K+1,IS)=(A(I,K+1,IS)-V)
      100   A(I,K,IS)=(A(I,K,IS)-P)
      C
      C     CONSERVATION DES TRANSFORMATIONS
      C
80          DO 120 I=1,N
            P=(X*VECT(I,K,IS))
            V=(Y*VECT(I,K+1,IS))
            P=(P+V)
            IF(LAST)GOTO 110
85          V=(Z*VECT(I,K+2,IS))
            P=(P+V)
            V=(P*R)
            VECT(I,K+2,IS)=(VECT(I,K+1,IS)-V)
      110    V=(P*Q)
90          VECT(I,K+1,IS)=(VECT(I,K+1,IS)-V)
            VECT(I,K  ,IS)=(VECT(I,K  ,IS)-P)
      120   CONTINUE
      130    CONTINUE
            RETURN
95          END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 QR

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	270	I	INTEGER		
0	IS	INTEGER		F.P.	301	J	INTEGER		
273	K	INTEGER			266	LAST	LOGICAL		
0	MAXL	INTEGER		F.P.	0	N	INTEGER		F.P.
0	NL	INTEGER		F.P.	267	NL2	INTEGER		
271	NL3	INTEGER			0	NU	INTEGER		F.P.
272	NUM1	INTEGER			0	P	REAL		F.P.
0	Q	REAL		F.P.	0	R	REAL		F.P.

```
      SUBROUTINE RACINE(A,N,MAXL,L,E1,E2,VECT,IS)
      DIMENSION A(MAXL,MAXL,3),VECT(MAXL,MAXL,3),E1(3),E2(3)
      COMMON/CHSIRQ/IMUL(3)
      REAL IMUL
C
      X=A(L+1,L+1,IS)
      Y=A(L,L,IS)
      W=(A(L,L+1,IS)*A(L+1,L,IS))
      P=(Y-X)
      P=(P/2.)
      Q=(P**2)
      Q=(Q+W)
      IF(Q.GE.0.)GOTO 5
      E1(IS)=(P+X)/IMUL(IS)
      E2(IS)=(SQRT(-Q))/IMUL(IS)
      RETURN
C
5     CONTINUE
      Z=(SQRT(Q))
      IF(P.LT.0.)GOTO 10
      Z=(P+Z)
      GOTO 20
10    Z=(P-Z)
20    CONTINUE
      IF(Z.EQ.0.)GOTO 30
      R=(-W/Z)
      GOTO 40
30    R=0.
40    CONTINUE
      IF(ABS((X+Z)).GE.ABS((X+R)))Z=R
      Y=(Y-X)
      Y=(Y-Z)
      X=-Z
      T=A(L,L+1,IS)
      U=A(L,L+1,IS)
      IF((ABS(Y)+ABS(U)).LE.(ABS(T)+ABS(X)))GOTO 60
      Q=U
      P=Y
      GOTO 70
60    Q=X
      P=T
70    R=P*P+Q*Q
      R=(SQRT(R))
      IF(R.GT.0.)GOTO 80
      E1(IS)=A(L,L,IS)/IMUL(IS)
      E2(IS)=A(L+1,L+1,IS)/IMUL(IS)
      A(L+1,L,IS)=0.
      RETURN
C
80    P=(P/R)
      Q=(Q/R)
      DO 90 J=L,N
      Z=A(L,J,IS)
      A(L,J,IS)=P*Z+Q*A(L+1,J,IS)
90    A(L+1,J,IS)=P*A(L+1,J,IS)-Q*Z
      L1=L+1
      DO 100 I=1,L1
```

```
                      Z=A(I,L,IS)
                      A(I,L,IS)=P*Z+Q*A(I,L+1,IS)
     60          100  A(I,L+1,IS)=P*A(I,L+1,IS)-Q*Z
                      DO 110 I=1,N
                      Z=VECT(I,L,IS)
                      VECT(I,L,IS)=P*Z+Q*VECT(I,L+1,IS)
                 110  VECT(I,L+1,IS)=P*VECT(I,L+1,IS)-Q*Z
     65               A(L+1,L,IS)=0.
                      E1(IS)=A(L,L,IS)/IMUL(IS)
                      E2(IS)=A(L+1,L+1,IS)/IMUL(IS)
                      RETURN
                      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 RACINE

VARIABLES	SN	TYPE		RELOCATION						
0	A	REAL	ARRAY	F.P.	0	E1	REAL	ARRAY	F.P.	
0	E2	REAL	ARRAY	F.P.	234	I	INTEGER			
0	IMUL	REAL	ARRAY	CHSIRQ	0	IS	INTEGER		F.P.	
232	J	INTEGER			0	L	INTEGER		F.P.	
233	L1	INTEGER			0	MAXL	INTEGER		F.P.	
0	N	INTEGER		F.P.	224	P	REAL			
225	Q	REAL			227	R	REAL			
230	T	REAL			231	U	REAL			
0	VECT	REAL	ARRAY	F.P.	223	W	REAL			
221	X	REAL			222	Y	REAL			
226	Z	REAL								

EXTERNALS	TYPE	ARGS
SQRT	REAL	1 LIBRARY

INLINE FUNCTIONS	TYPE	ARGS
ABS	REAL	1 INTRIN

STATEMENT LABELS

36	5	42	10	44	20
47	30	50	40	76	60
101	70	123	80	0	90
0	100	0	110		

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
136	90	J	52 55	4B	INSTACK
155	100	I	57 60	4B	INSTACK
175	110	I	61 64	4B	INSTACK

COMMON BLOCKS	LENGTH
CHSIRQ	3

II.4.4 - Résultats numériques

Reprenons le programme d'appel de II.3.6 testant la matrice de II.1.6. Si nous comparons avec les résultats obtenus en II.1.6, il n'y a pas de différences notables ; seules, quelques petites variations existent, dues au caractère aléatoire de l'erreur numérique, et donc au nombre d'itérations effectuées non parfaitement identiques dans les deux cas. Pour vérifier l'exactitude des résultats numériques obtenus, nous avons fait un deuxième test en double précision. Les résultats obtenus avec calcul du nombre de chiffres significatifs nous permettent aussi de vérifier ceux obtenus en simple précision.

Nous observons alors que la précision exacte est de très peu supérieure à celle trouvée. En fait, comme le souligne Maillé |25|, choisir la partie entière de $C = \mathrm{Log}_{10} \frac{|\bar{R}|}{\delta}$ fait apparaître des discontinuités.

Soit par exemple $\delta = 0.01$ et $\bar{R} = 10$. Nous obtiendrons $C = 3$. Si $\bar{R} = 9.99$ nous obtiendrons $C = 2.9995655$ qui sera donc tronqué à 2.

Il est donc préférable d'arrondir à la valeur entière la plus proche, c'est-à-dire de rajouter 0.5 avant de tronquer. Mais, si nous désirons un intervalle de confiance à 5 %, nous avons vu que pour seulement trois valeurs de R il nous fallait ôter 0.69..., or $\mathrm{Log}_{10} \frac{|\bar{R}|}{\delta}$ est proche de $\mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} - 0.69 + 0.5$. Donc, en procédant comme nous l'avons fait, nous sommes à peu près dans notre intervalle de confiance. Ce qui est une bonne sécurité.

Nous avons, bien sûr, triplé notre volume de calcul pour chaque itération. Mais nous avons constaté une diminution du nombre d'itérations lorsque nous prenions des matrices bien conditionnées et de taille plus importante.

NO	PARTIE REELLE	CHIFF.SIGN.	PARTIE IMAGINAIRE	CHIFF.SIGN.
1	.45946781371747E+02	12	0.	15
2	.31913919887477E+02	12	0.	15
3	.22141483257827E+02	12	0.	15
4	.14899173038189E+02	12	0.	15
5	.95138427238770E+01	12	0.	15
6	.56326922741220E+01	12	0.	15
7	.30237870566649E+01	12	0.	15
8	.14687177518787E+01	9	0.	15
9	.68086603952477E+00	6	0.	15
10	.33071182132329E+00	4	0.	15
11	.17748112279988E+00	2	0.	15
12	.10573908610225E+00	1	0.	15
13	.67259334183086E-01	1	0.	15
14	.48953845881071E-01	0	0.	15
15	.25909307339195E-01	0	0.	15
16	.22682080952330E-01	0	0.	15

Tableau IV

NO	PARTIE REELLE	CHIFF.SIGN.	PARTIE IMAGINAIRE	CHIFF.SIGN.
1	.45946781371779E+02	26	0.	15
2	.31913919887504E+02	26	0.	15
3	.22141483257850E+02	26	0.	15
4	.14899173038203E+02	26	0.	15
5	.95138427238851E+01	27	0.	15
6	.56326922741261E+01	26	0.	15
7	.30237870566722E+01	21	0.	15
8	.14687177523947E+01	21	0.	15
9	.68086601279894E+00	20	0.	15
10	.33071111862637E+00	18	0.	15
11	.17753499593676E+00	17	0.	15
12	.10510999908475E+00	15	0.	15
13	.67117819051831E-01	14	0.	15
14	.45164092592824E-01	14	0.	15
15	.31334289346905E-01	10	0.	15
16	.21764310147906E-01	10	0.	15

Tableau V

II.5 - CONCLUSIONS

Les méthodes de perturbation et de multiplication apportent une plus grande stabilité de l'algorithme QR, permettent un nombre d'itérations optimal (gain d'itérations non négligeable sur des matrices d'ordre important), et enfin permettent de vérifier directement la validité des résultats obtenus.

Les méthodes de perturbation sont générales à tout algorithme et donnent des résultats satisfaisants. Maintenant, chaque fois que nous pouvons nous ramener à la méthode de multiplication pour abaisser les coûts machines, cette méthode sera choisie. De plus, dans certains cas, cette méthode devient triviale à mettre en oeuvre.

Exemple : désignons par AHS un programme de mise en forme Hessenberg supérieure pour une matrice quelconque. Soit A, la matrice quelconque, B, la matrice de forme de Hessenberg supérieure, et écrivons B = AHS(A). B est semblable à A, et les transformations effectuées sont encore des transformations unitaires d'où k.B = AHS(k.A). Il est alors possible de constituer trois matrices $B_i = AHS(k_i.A)/k_i$ que nous pouvons inclure dans le programme CHSIQR avec IARG = 1. Nous tiendrons compte alors des erreurs numériques liées à la mise en forme de Hessenberg supérieure.

D'autres applications peuvent être faites en amont ou en aval de l'algorithme QR en utilisant le même procédé.

Si nous sommes maintenant en mesure de détecter les erreurs numériques, nous ne pouvons cependant pas éviter leurs propagations.

Voyons maintenant comment se comportent d'autres algorithmes sur notre matrice mal conditionnée.

III - RESOLUTION DU POLYNOME CARACTERISTIQUE

III.1 - INTRODUCTION

Lorsque l'on étudie les méthodes de calcul de valeurs propres, la première idée est celle du calcul du polynôme caractéristique et de sa résolution. Sur ordinateur, cette méthode est inemployée, car réputée instable et coûteuse.

1) Est-elle vraiment instable ?

2) La détection des cas instables ne résout-elle pas en partie le problème ?

3) La nouvelle génération des ordinateurs tels que le CRAY-1 ou le CYBER 205 introduisant la "vectorisation" (augmentation importante du débit des calculs lorsque les calculs sont effectués sur des vecteurs longs) ne va-t-elle pas remettre au goût du jour des algorithmes désuets ?

L'algorithme QR est, a priori, mal vectorisable car la majorité du temps calcul est effectuée sur les scalaires p_m, q_m, r_m (cf. (2.1.19)).

Nous allons donc tester cette méthode sur la matrice définie en II.1.6 et nous choissons la méthode de Faddeev |8| pour sa simplicité de mise en oeuvre et son grand pouvoir de vectorisation. Pour sa résolution, nous choisirons l'algorithme de Bairstow stabilisé |38| afin d'obtenir la meilleure résolution possible.

III.2 - LA METHODE DE FADDEEV

III.2.1 - Principe

Cette méthode, décrite aussi dans |10|, permet d'obtenir, par un algorithme direct, les coefficients $p_1,\ldots,p_n$ du polynôme caractéristique :

$$\Delta(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \ldots - p_{n-1} \lambda^1 - p_n \qquad (3.2.1)$$

Les matrices $B_1 \ldots B_{n-1}$ calculées au cours de cet algorithme sont les matrices coefficients de la matrice adjointe au polynôme caractéristique : $B(\lambda)$. Elles peuvent être utilisées pour le calcul des vecteurs propres après résolution du polynôme.

L'algorithme est décrit par les formules de récurrence suivantes :

$$\begin{array}{lll} A_1 = A & , \; p_1 = \mathrm{tr}\, A_1 & , \; B_1 = A_1 - p_1 E \\ A_2 = A\, B_1 & , \; p_2 = \tfrac{1}{2}\, \mathrm{tr}\, A_2 & , \; B_2 = A_2 - p_2 E \\ \ldots & \ldots & \ldots \\ A_n = A\, B_{n-1} & , \; p_n = \dfrac{1}{n}\, \mathrm{tr}\, A_n & , \; B_n = A_n - p_n E \end{array} \qquad (3.2.2)$$

A est la matrice initiale, E la matrice unité, trA est la trace de la matrice A, soit :

$$\mathrm{tr}\, A = \sum_{i=1}^{n} a_{i,i} \qquad (3.2.3)$$

Nous avons les propriétés suivantes :

- Les coefficients p_i sont bien les coefficients du polynôme caractéristique (3.2.1).
- Les matrices B_i sont les matrices coefficients de la matrice adjointe $B(\lambda)$.
- $B_n = 0$: matrice nulle.
- $p_n = (-1)^{n-1} \det(A)$ (3.2.4)
- Enfin, si A est régulière, de $A_n - p_n E = 0$ et $A_n = AB_{n-1}$, nous tirons $AB_{n-1} = p_n E$, soit :

$$A^{-1} = \frac{1}{p_n} B_{n-1} \qquad (3.2.5)$$

Ces propriétés sont une conséquence du théorème suivant |10, p. 84| et de la formule des coefficients de Newton.

Théorème 3.2.1 :

Si $\lambda_1, \ldots, \lambda_n$ sont les valeurs propres d'une matrice A, si $g(\mu)$ est un polynôme, alors $g(\lambda_1), \ldots, g(\lambda_n)$ sont les valeurs propres de $g(A)$.

En particulier, λ_1^k, λ_2^k, ... λ_n^k sont les valeurs propres de A^k et nous obtenons :

$$\text{tr } A^k = s_k = \sum_{i=1}^{n} \lambda_i^k \quad , k = 1,\ldots,n \tag{3.2.6}$$

Par ailleurs, les sommes s_k sont liées par la formule des coefficients de Newton :

$$k\, p_k = s_k - p_1\, s_{k-1} - \ldots - p_{k-1}\, s_1 \tag{3.2.7}$$

où p_k sont les coefficients du polynôme caractéristique (3.2.1).

III.2.2 - Utilisation de la méthode de multiplication

Nous nous trouvons ici dans un cas simple de mise en oeuvre de la méthode de multiplication.

Si nous prenons comme matrice initiale la matrice $A' = k.A$ où k est un scalaire, nous obtiendrons des coefficients p'_i tels que :

$$p'_i = k^i . p_i \tag{3.2.8}$$

Néanmoins, par souci de stabilité et afin de ne pas traîner des termes en k^i, nous avons modifié l'algorithme de la façon suivante :

$$\begin{array}{lll}
A'_1 = k.A & , \; p'_1 = \text{tr } A'_1 & , \; B'_1 = A'_1 - p'_1 E \\
A'_2 = A.B'_1 & , \; p'_2 = \tfrac{1}{2} \text{ tr } A'_2 & , \; B'_2 = A'_2 - p'_2 E \\
\ldots & \ldots & \ldots \\
A'_n = A.B'_{n-1} & , \; p'_n = \frac{1}{n} \text{ tr } A'_n & , \; B'_n = A'_n - p'_n E
\end{array} \tag{3.2.9}$$

qui se ramène à l'algorithme (3.2.2) par :

$$\begin{array}{lll}
A'_1 = k.A_1 & , \; p'_1 = k.p_1 & , \; B'_1 = k.B_1 \\
A'_2 = k.A_2 & , \; p'_2 = k.p_2 & , \; B'_2 = k.B_2 \\
\ldots & \ldots & \ldots \\
A'_n = k.A_n & , \; p'_n = k.p_n & , \; B'_n = k.B_n = 0
\end{array} \tag{3.2.10}$$

III.2.3 - Mise en oeuvre

Les trois calculs des coefficients du polynôme avec des coefficients multiplicatifs de A différents peuvent être faits ici l'un après l'autre, puisqu'il s'agit d'une méthode directe.

Seul, un test sur le coefficient p_n s'avère nécessaire avant le calcul de la matrice inverse, ce dernier calcul étant si peu coûteux qu'il serait mal venu de ne pas l'effectuer.

Nous ferons appel pour le produit matriciel au logiciel de base de toute bibliothèque mathématique. D'autant que sur le CRAY-1 nous pouvons trouver ce sous-programme comme une fonction système entièrement vectorisée, elle existe aussi sur n'importe quel arrêt-processeur.

Enfin, nous baptiserons ce programme POLCAR.

III.2.4 - Notice d'utilisation et organigramme simplifié du programme POLCAR

Objet

Calcul des coefficients du polynôme caractéristique $P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \ldots - p_{n-1} \lambda - p_n$ pour une matrice A quelconque. Le nombre de chiffres significatifs sur ses coefficients est donné. La matrice inverse est calculée si le déterminant $(-1)^{n-1} p_n$ est numériquement non nul.

Appel

```
CALL POLCAR(A,B,C,P,CP,IA,N)
```

Arguments d'entrée

A — Tableau bidimensionné dont la première dimension est IA et la seconde au moins égale à N. A contient la matrice dont nous voulons calculer le polynôme caractéristique. A n'est pas détruite.

C — Tableau de travail de mêmes dimensions que A.

IA — Première dimension de A,B,C, et P.

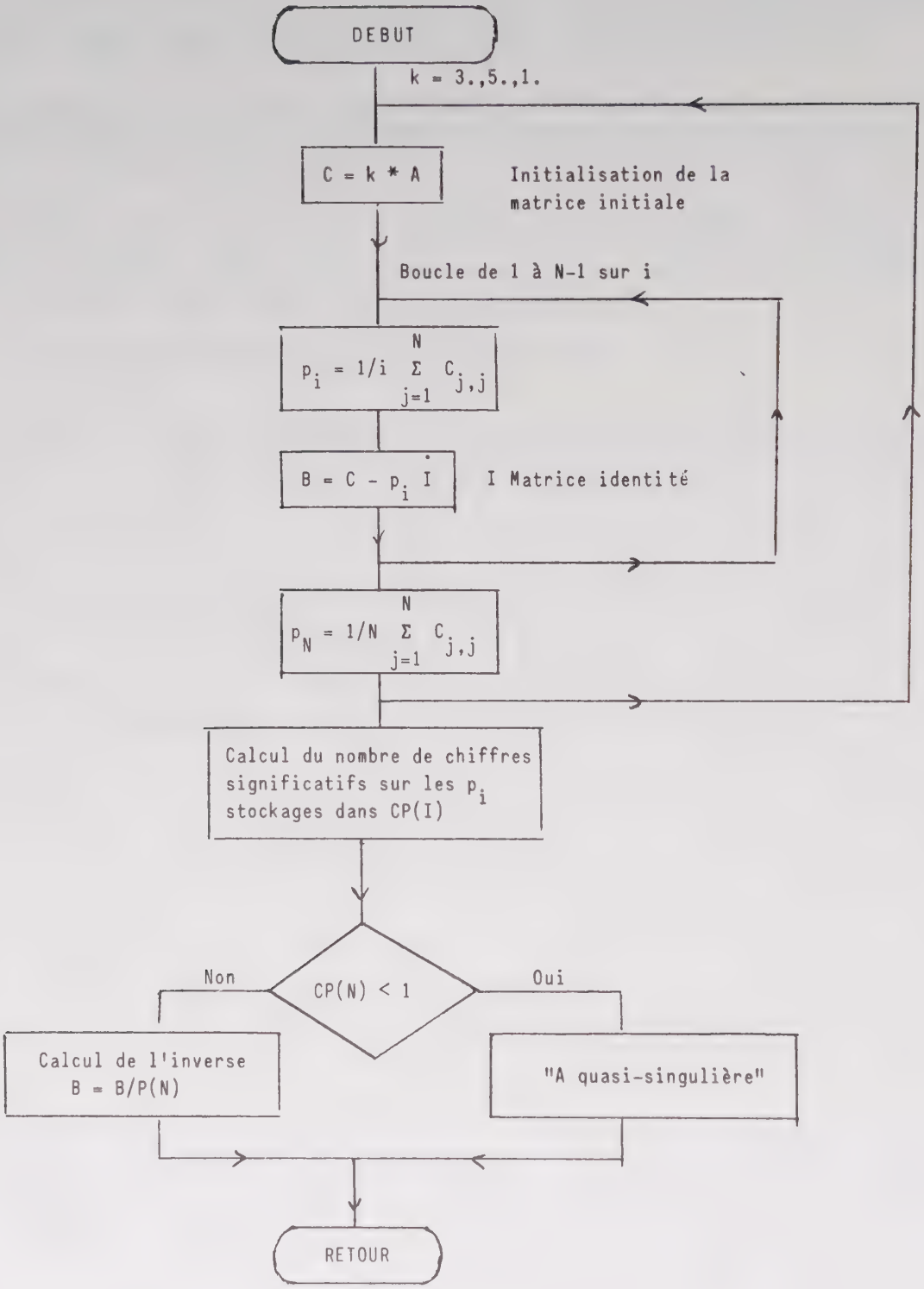
N — Ordre de la matrice A.

Arguments de sortie

B — Tableau de mêmes dimensions que A, contenant la matrice inverse.

P — Tableau bidimensionné dont la première dimension est IA et la seconde 3. Ce sont 3 résultats numériques du polynôme caractéristique. Ils sont en théorie identiques, ils ne le sont pas en cas d'instabilité numérique.
P(I,1) correspond au coefficient p_i cité plus haut pour I = 1,N.

CP — Tableau monodimensionné, contient le nombre de chiffres décimaux significatifs pour chacun des coefficients du polynôme caractéristique calculés. CP(I) correspond encore à p_i. CP est réel.


DEBUT
k = 3.,5.,1.
C = k * A
Initialisation de la matrice initiale
Boucle de 1 à N-1 sur i
$p_i = 1/i \sum_{j=1}^{N} C_{j,j}$
B = C - p_i I
I Matrice identité
$p_N = 1/N \sum_{j=1}^{N} C_{j,j}$
Calcul du nombre de chiffres significatifs sur les p_i stockages dans CP(I)
CP(N) < 1
Non
Oui
Calcul de l'inverse B = B/P(N)
"A quasi-singulière"
RETOUR

III.2.5 - Programme FORTRAN POLCAR

```
      SUBROUTINE POLCAR(A,B,C,P,CP,IA,N)                                00000400
      DIMENSION A(IA,1),B(IA,1),C(IA,1),P(IA,3),CP(1),T(3),MUL(3)       00000410
      LOGICAL RETOUR                                                    00000420
      REAL MUL                                                          00000430
      COMMON/WXTST/CMIN,CMAX                                            00000440
      DATA MUL/3.,5.,1. /                                               00000450
      N1=N-1                                                            00000460
      CMIN=1.                                                           00000470
      CMAX=15.                                                          00000480
      NS=3                                                              00000490
      DO 1 L=1,NS                                                       00000500
      DO 2 I=1,N                                                        00000510
      DO 2 J=1,N                                                        00000520
    2 C(I,J)=A(I,J)*MUL(L)                                              00000530
      DO 3 K=1,N1                                                       00000540
      P(K,L)=0                                                          00000550
      DO 4 I=1,N                                                        00000560
    4 P(K,L)=P(K,L)+C(I,I)                                              00000570
      P(K,L)=P(K,L)/K                                                   00000580
      DO 5 I=1,N                                                        00000590
      DO 6 J=1,N                                                        00000600
    6 B(I,J)=C(I,J)                                                     00000610
    5 B(I,I)=     (B(I,I)-P(K,L))                                       00000620
      IER=0.                                                            00000630
      CALL ALPRO(IA,IA,N,N,N,A,B,C,IER)                                 00000640
    3  CONTINUE                                                         00000650
      P(N,L)=0                                                          00000660
      DO 8 I=1,N                                                        00000670
    8 P(N,L)=     (P(N,L)+C(I,I))                                       00000680
      P(N,L)=P(N,L)/N                                                   00000690
    1 CONTINUE                                                          00000700
      DO 9 K=1,N                                                        00000710
      DO 13 L=1,NS                                                      00000720
      P(K,L)=P(K,L)/MUL(L)                                              00000730
   13 T(L)=P(K,L)                                                       00000740
      CALL WXTEST(T,3,CP(K),RM,RETOUR)                                  00000750
    9 CONTINUE                                                          00000760
      C                                                                 00000770
      C     CALCUL DE LA MATRICE INVERSE                                00000780
      C                                                                 00000790
      IF(RETOUR)GOTO 12                                                 00000800
      IF(RM.EQ.0.)GOTO 12                                               00000810
      DO 10 I=1,N                                                       00000820
      DO 10 J=1,N                                                       00000830
   10 B(I,J)=B(I,J)/RM                                                  00000840
      RETURN                                                            00000850
   12 PRINT 11                                                          00000860
   11 FORMAT(* MATRICE QUASI-SINGULIERE,MATRICE INVERSE NON CALCULEE*)  00000870
      RETURN                                                            00000880
      END                                                               00000890
```

SYMBOLIC REFERENCE MAP (R=1)

```
      1              SUBROUTINE ALPRO(IA,IA2,N,N2,N3,A,B,C,IER)                00000900
                     DIMENSION A(IA,1),B(IA,1),C(IA,1)                         00000910
                     DO 1 I=1,N                                                00000920
                     DO 1 J=1,N                                                00000930
      5              C(I,J)=0.                                                 00000940
                     DO 1 K=1,N                                                00000950
                1    C(I,J)=A(I,K)*B(K,J)+C(I,J)                               00000960
                     RETURN                                                    00000970
                     END                                                       00000980
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3 ALPRO

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	B	REAL	ARRAY	F.P.
0	C	REAL	ARRAY	F.P.	40	I	INTEGER		
0	IA	INTEGER		F.P.	0	IA2	INTEGER	*UNUSED	F.P.
0	IER	INTEGER	*UNUSED	F.P.	41	J	INTEGER		
42	K	INTEGER			0	N	INTEGER		F.P.
0	N2	INTEGER	*UNUSED	F.P.	0	N3	INTEGER	*UNUSED	F.P.

STATEMENT LABELS
    0 1

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
14	1	I	3 7	21B	NOT INNER
17	1	J	4 7	13B	NOT INNER
25	1	K	6 7	3B	INSTACK

STATISTICS
  PROGRAM LENGTH 47B 39
        55000B SCM USED

```
      SUBROUTINE POLCAR(A,B,C,P,CP,IA,N)                                00000400
      DIMENSION A(IA,1),B(IA,1),C(IA,1),P(IA,3),CP(1),T(3),MUL(3)       00000410
      LOGICAL RETOUR                                                    00000420
      REAL MUL                                                          00000430
      COMMON/WXTST/CMIN,CMAX                                            00000440
      DATA MUL/3.,5.,1. /                                               00000450
      N1=N-1                                                            00000460
      CMIN=1.                                                           00000470
      CMAX=15.                                                          00000480
      NS=3                                                              00000490
      DO 1 L=1,NS                                                       00000500
      DO 2 I=1,N                                                        00000510
      DO 2 J=1,N                                                        00000520
    2 C(I,J)=A(I,J)*MUL(L)                                              00000530
      DO 3 K=1,N1                                                       00000540
      P(K,L)=0                                                          00000550
      DO 4 I=1,N                                                        00000560
    4 P(K,L)=P(K,L)+C(I,I)                                              00000570
      P(K,L)=P(K,L)/K                                                   00000580
      DO 5 I=1,N                                                        00000590
      DO 6 J=1,N                                                        00000600
    6 B(I,J)=C(I,J)                                                     00000610
    5 B(I,I)=     (B(I,I)-P(K,L))                                       00000620
      IER=0.                                                            00000630
      CALL ALPRO(IA,IA,N,N,N,A,B,C,IER)                                 00000640
    3  CONTINUE                                                         00000650
      P(N,L)=0                                                          00000660
      DO 8 I=1,N                                                        00000670
    8 P(N,L)=     (P(N,L)+C(I,I))                                       00000680
      P(N,L)=P(N,L)/N                                                   00000690
    1 CONTINUE                                                          00000700
      DO 9 K=1,N                                                        00000710
      DO 13 L=1,NS                                                      00000720
      P(K,L)=P(K,L)/MUL(L)                                              00000730
   13 T(L)=P(K,L)                                                       00000740
      CALL WXTEST(T,3,CP(K),RM,RETOUR)                                  00000750
    9 CONTINUE                                                          00000760
      C                                                                 00000770
      C     CALCUL DE LA MATRICE INVERSE                                00000780
      C                                                                 00000790
      IF(RETOUR)GOTO 12                                                 00000800
      IF(RM.EQ.0.)GOTO 12                                               00000810
      DO 10 I=1,N                                                       00000820
      DO 10 J=1,N                                                       00000830
   10 B(I,J)=B(I,J)/RM                                                  00000840
      RETURN                                                            00000850
   12 PRINT 11                                                          00000860
   11 FORMAT(* MATRICE QUASI-SINGULIERE,MATRICE INVERSE NON CALCULEE*)  00000870
      RETURN                                                            00000880
      END                                                               00000890
```

SYMBOLIC REFERENCE MAP (R=1)

III.2.6 - Résultats numériques

Nous testons de nouveau notre algorithme sur la matrice 16x16 définie en II.1.6 et sur un matériel CDC 7600.

```
      PROGRAM TEST1(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)
      DIMENSION A(50,50),B(50,50),C(50,50),P(50,3),CP(50)
      DIMENSION COE(51),RREEL(51),RIMAG(51)
      IA=50
      N=20
      PRINT 8
      DO 1 I=1,N
      DO 2 J=1,I
    2 A(I,J)=0.
      DO 3 J=I,N
    3 A(I,J)=N-J+1
      IF(I.EQ.1)GOTO 1
      A(I,I-1)=N-I+1
    1 PRINT 7,(A(I,J),J=1,N)
    7 FORMAT(1X,20(F4.0,1X))
    8 FORMAT(1H1)
      CALL POLCAR(A,B,C,P,CP,IA,N)
       PRINT 8
      PRINT 9
    9 FORMAT(* CALCUL DES COEFFICIENTS DU POLYNOME CARACTERISTIQUE*//,
     1 *   NO   COEFFICIENTS          CH.SIGN.*/)
      DO 4 I=1,N
      NCH=CP(I)
    4 PRINT 5,I,P(I,3),NCH
    5 FORMAT(1X,I4,3X,E21.14,5X,I3)
      STOP
      END
```

Les résultats fournis nous montrent que les calculs ont été effectués avec 15 chiffres décimaux significatifs exacts sur les coefficients du polynôme caractéristique !

Une instabilité numérique est montrée néanmoins sur la même matrice, à partir des dimensions 20x20.

La précision sur le dernier coefficient p_n passe alors brusquement de 15 chiffres décimaux significatifs à 0. Il est clair que dans ce dernier cas, il ne peut y avoir résolution du polynôme.

Par contre, sur la matrice 16x16, il n'y a aucune erreur numérique. La précision sur les valeurs propres va donc dépendre du programme de résolution de ce polynôme.

Résultats pour la matrice 16x16 :

CALCUL DES COEFFICIENTS DU POLYNOME CARACTERISTIQUE

NO	COEFFICIENTS	CH.SIGN.
1	.13600000000000E+03	15
2	-.72600000000000E+04	15
3	.19712000000000E+06	15
4	-.29584100000000E+07	15
5	.25109448000000E+08	15
6	-.11952740800000E+09	15
7	.31100212000000E+09	15
8	-.42965851500000E+09	15
9	.31100212000000E+09	15
10	-.11952740800000E+09	15
11	.25109448000000E+08	15
12	-.29584100000000E+07	15
13	.19712000000000E+06	15
14	-.72600000000000E+04	15
15	.13600000000000E+03	15
16	-.10000000000000E+01	15

Tableau VI

Résultats pour la matrice 20x20 :

CALCUL DES COEFFICIENTS DU POLYNOME CARACTERISTIQUE

NO	COEFFICIENTS	CH.SIGN.
1	.21000000000000E+03	15
2	-.18145000000000E+05	15
3	.84417000000000E+06	15
4	-.23270535000000E+08	15
5	.39496052400000E+09	15
6	-.41681244300000E+10	15
7	.27129228660000E+11	15
8	-.10669791268500E+12	15
9	.24678938063000E+12	15
10	-.32750490587100E+12	15
11	.24678938063000E+12	12
12	-.10669791268467E+12	11
13	.27129228661151E+11	8
14	-.41681242940129E+10	6
15	.39496786725606E+09	3
16	-.22881015141966E+08	0
17	.21773189244956E+08	0
18	.11375251190887E+10	0
19	.62378927707814E+11	0
20	.3444103991291 6E+13	0

Tableau VII

III.3 - RESOLUTION DU POLYNOME CARACTERISTIQUE CALCULE

III.3.1 - Utilisation de la méthode de Bairstow stabilisée

Nous trouverons le principe de cet algorithme et le programme FORTRAN correspondant dans |38|. Cet algorithme diffère de l'algorithme de Bairstow classique (faisant appel à la méthode de Raphson-Newton) par un test d'arrêt optimal sur les itérations Raphson-Newton limitant la propagation des erreurs, plus un contrôle de la solution avec amélioration du résultat. Pour cela, il utilise aussi la méthode de permutation-perturbation pour le test optimal et le contrôle de la solution.

Son utilisation va donc nous permettre d'obtenir la meilleure précision possible sur les racines de ce polynôme et nous donner directement le nombre de chiffres décimaux significatifs sur les racines, c'est-à-dire sur les valeurs propres.

Il n'est prévu d'entrer dans ce programme (programme BAIPR) que des coefficients à valeurs numériques exactes, ce qui est notre cas ici. Pour une utilisation plus générale, il faudrait alors le modifier.

III.3.2 - Résultats numériques

L'appel au programme BAIPR fait suite à celui du programme POLCAR.

```
      PROGRAM TEST1(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)
      DIMENSION A(50,50),B(50,50),C(50,50),P(50,3),CP(50)
      DIMENSION COE(51),RREEL(51),RIMAG(51)
      IA=50
      N=16
      PRINT 8
      DO 1 I=1,N
      DO 2 J=1,I
    2 A(I,J)=0.
      DO 3 J=I,N
    3 A(I,J)=N-J+1
      IF(I.EQ.1)GOTO 1
      A(I,I-1)=N-I+1
    1 PRINT 7,(A(I,J),J=1,N)
    7 FORMAT(1X,20(F4.0,1X))
    8 FORMAT(1H1)
      CALL POLCAR(A,B,C,P,CP,IA,N)
       PRINT 8
      PRINT 9
    9 FORMAT(* CALCUL DES COEFFICIENTS DU POLYNOME CARACTERISTIQUE*//,
     1 *   NO   COEFFICIENTS          CH.SIGN.*/)
      DO 4 I=1,N
      NCH=CP(I)
    4 PRINT 5,I,P(I,3),NCH
    5 FORMAT(1X,I4,3X,E21.14,5X,I3)
      NP1=N+1
      IARG=0
      COE(1)=1.
      DO 6 I=1,N
    6 COE(I+1)=-P(I)
      PRINT 8
      CALL BAIRPR(NP1,COE,RREEL,RIMAG,IARG)
      STOP
      END
```

Les résultats numériques donnés page suivante sont très encourageants et peuvent être comparés avec ceux obtenus en double précision de II.4.4. Certains d'entre eux sont même meilleurs !

Tableau VIII

RACINE NUMERO 1			
PARTIE REELLE= .217643101463146E-01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.10000000E+01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 2			
PARTIE REELLE= .313342893484716E-01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.18612140E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 3			
PARTIE REELLE= .451640925928583E-01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.72422193E-01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 4			
PARTIE REELLE= .671178190518407E-01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.81636674E-01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 5			
PARTIE REELLE= .1051099990847?8E+00	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.23997354E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 6			
PARTIE REELLE= .177534995936687E+00	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.12058162E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 7			
PARTIE REELLE= .330711118626406E+00	NBRE DE CHIFFRES EXACTS 13	RES. NOR.=	.22974523E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 8			
PARTIE REELLE= .680866012798635E+00	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.56286887E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 9			
PARTIE REELLE= .146871775239619E+01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.79460145E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 10			
PARTIE REELLE= .302378705666592E+01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.12127195E+01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 11			
PARTIE REELLE= .563269227414341E+01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.53733568E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 12			
PARTIE REELLE= .951384272386366E+01	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.38404664E-01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 13			
PARTIE REELLE= .148991730381915E+02	NBRE DE CHIFFRES EXACTS 11	RES. NOR.=	.46330608E-01
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 14			
PARTIE REELLE= .221414832579027E+02	NBRE DE CHIFFRES EXACTS 11	RES. NOR.=	.29882492E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 15			
PARTIE REELLE= .319139198874518E+02	NBRE DE CHIFFRES EXACTS 11	RES. NOR.=	.41239838E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01
RACINE NUMERO 16			
PARTIE REELLE= .459467813718031E+02	NBRE DE CHIFFRES EXACTS 12	RES. NOR.=	.24175343E+00
PARTIE IMAGINAIRE 0.	NBRE DE CHIFFRES EXACTS 14	RES. NOR.=	.10000000E+01

III.4 - CONCLUSIONS

L'exemple de notre matrice 16x16 montre que les résultats obtenus par cette méthode peuvent être meilleurs dans certains cas que ceux obtenus par la méthode QR.

Nous observons avec l'exemple 20x20 que cet algorithme possède aussi ses limites. Néanmoins, à ces dimensions, plus aucun algorithme de calcul de valeurs propres ne donne des résultats corrects en arithmétique double précision sur la même machine.

Par ailleurs, la "vectorisation" des produits matriciels peut rendre compétitif cet algorithme.

Enfin, dans |10|, nous trouverons une méthode pour obtenir les vecteurs propres a posteriori.

IV - AMELIORATION DES VALEURS PROPRES ET DES VECTEURS PROPRES

IV.1 - INTRODUCTION

Par une méthode ou une autre, nous avons obtenu des résultats numériques plus ou moins précis sur les valeurs propres d'une matrice.

La connaissance de la précision de ces résultats nous incitera éventuellement à augmenter la précision des calculs. Cependant, il y a une limite à l'augmentation de la précision. Ceci nous oblige à recalculer toutes les valeurs propres. Enfin, quelle précision est nécessaire pour pouvoir affirmer que ce calcul est satisfaisant ?

Des algorithmes d'amélioration de la précision existent. Mais il faudrait pouvoir tenir compte du mauvais conditionnement de la valeur propre pour la matrice associée. Il n'est pas sûr que dans le cas de résultats instables, les algorithmes d'amélioration de la précision actuels soient plus stables que l'algorithme de calcul lui-même.

Il faut donc construire un algorithme d'amélioration de la précision qui tient compte de l'erreur numérique issu de ses propres calculs.

Ilest évident que nous ne pourrons rien si les valeurs propres ne sont pas suffisamment séparées, c'est-à-dire si l'imprécision est supérieure à la distance entre deux valeurs propres voisines.

Enfin, si les vecteurs propres sont obtenus par la méthode QR, alors la précision des vecteurs propres dépend de la précision des valeurs propres. Nous verrons qu'il faut améliorer valeurs propres et vecteurs propres indépendamment et qu'un résidu négligeable ($r = ||R||_2 = ||(A- \lambda I)V||_2$) n'est pas un critère suffisant pour pouvoir affirmer qu'une valeur propre et son vecteur propre associé ont un grand nombre de chiffres décimaux significatifs.

IV.2 - AMELIORATION DES VALEURS PROPRES

IV.2.1 - Recherche de la singularité numérique

Pour que λ soit une valeur propre de la matrice A, il faut et il suffit que $(A-\lambda I)$ soit singulière, I étant la matrice identité.

Ceci est encore équivalent au fait que le déterminant de $(A-\lambda I)$ soit nul (noté $\det(A-\lambda I) = 0$).

Même si $(A-\lambda I)$ est théoriquement singulière, il est rare que le déterminant calculé soit nul en machine. Dans ce cas, il est plutôt égal à l'erreur propagée par les erreurs d'arrondi ou de troncature.

Nous nous référons alors à |37| pour affirmer qu'une matrice est numériquement singulière si le nombre de chiffres significatifs de son déterminant est nul. De même, nous considérerons qu'une valeur propre est numériquement satisfaisante lorsque $\det(A-\lambda I)$ n'a plus aucun chiffre significatif.

Il est clair qu'avec une telle définition, le critère "satisfaisant" est lié à l'algorithme de calcul du déterminant. Néanmoins, la méthode de Gauss avec recherche du pivot maximal par colonne peut être la méthode de référence. Cette méthode est celle utilisée dans tous les logiciels de base, car elle nécessite un nombre d'opérations élémentaires minimal. Ce n'est pas obligatoirement la méthode la plus stable, mais c'est celle que nous utiliserons a priori.

Ayant un critère de choix pour une valeur propre satisfaisante, il nous faut alors trouver un algorithme pour atteindre cette valeur, et ce aussi bien pour $\lambda \in \mathbb{C}$ que pour $\lambda \in \mathbb{R}$. Pour cela, nous chercherons le zéro local de la fonction $f(\lambda) = |\det(A-\lambda I)|^2$, nous ramenant ainsi à la recherche d'un minimum local d'une fonction quadratique (nous le démontrerons).

La dérivée de la fonction f étant mal aisée à calculer, nous procéderons à une recherche dichotomique sur $\mathbb{C}$ ou sur $\mathbb{R}$. Si pour $\lambda \in \mathbb{R}$ cette recherche est triviale (en supposant que les coefficients de A soient tous réels), le problème est plus difficile pour $\lambda \in \mathbb{C}$. Nous utiliserons alors un algorithme construit à partir de la méthode de Rosenbrock |31| dont nous trouverons la description ainsi que le programme FORTRAN:OPTIM de cette version dans |36|.

IV.2.2 - Utilisation du programme OPTIM

Ce programme, décrit dans |36|, recherche un extremum d'une fonction à plusieurs variables, sans calculer les dérivées. Il nous est donc très utile pour chercher le zéro de $f(\lambda) = |\det(A-\lambda I)|^2$. Mais il est clair que cette convergence théorique ne pourra s'accompagner d'une convergence numérique que si nous regardons à chaque étape de calcul le nombre de chiffres décimaux significatifs de la valeur numérique de $f(\lambda)$, et que zéro chiffre décimal significatif soit pour nous un critère de zéro numérique.

Pour éviter une divergence de l'algorithme, il faudra aussi tenir compte de ce bruit numérique dans notre étape de recherche.

Nous allons donc modifier le programme OPTIM et en redonner le principe.

IV.2.3 - Principe du programme OPTIM modifié

L'idée directrice de cet algorithme est de former une suite $f(\lambda_j)$ décroissante et telle que la différence $f(\lambda_j) - f(\lambda_{j+1})$ soit quasi-maximum à chaque itération.

Si nous décomposons λ en $\lambda = \lambda_R + i\lambda_I$ (partie réelle et partie imaginaire), nous pouvons considérer $f(\lambda)$ comme une fonction de $\mathbb{R}^2 \longrightarrow \mathbb{R}$, nous noterons alors indifféremment $f(\lambda)$ ou $f(\lambda_R,\lambda_I)$ ou encore $f(\lambda_1,\lambda_2)$ avec $\lambda_1 = \lambda_R$ et $\lambda_2 = \lambda_I$.

Nous partons d'une solution initiale approchée $\lambda^0 = \lambda$, soit $(\lambda_1^0, \lambda_2^0) = (\lambda_1, \lambda_2)$ et d'un pas d'incrémentation initial (ou plage d'erreurs) $(\Delta\lambda_1^0, \Delta\lambda_2^0) = (\Delta\lambda_1, \Delta\lambda_2)$.

A l'étape j, nous cherchons le point $X^j = (x_1^j, x_2^j)$ appartenant à un ensemble de 9 points dont les valeurs possibles de chacune des coordonnées sont

$$x_i^j = \lambda_i^j + \Delta\lambda_i^j,\ x_i^j = \lambda_i^j,\ x_i^j = \lambda_i^j - \Delta\lambda_i^j \quad \text{pour } i = 1,2 \qquad (4.2.1)$$

et tel que $f(X^j)$ soit la valeur minimale de ces 9 points.

Plusieurs cas peuvent se présenter :

a) $f(X^j)$ n'a aucun chiffre significatif, alors $f(X^j)$ doit être considéré comme un zéro numérique et X^j correspond à notre valeur propre cherchée. On notera $f(X^j) \equiv 0$.

b) $X^j \neq \lambda^j$ et $f(X^j) \not\equiv 0$: nous cherchons alors un point X'^j dans la direction $\overrightarrow{\lambda^j X^j}$ et au-delà de X^j, par une étape appelée accélération, exposée plus loin.

c) $X^j = \lambda^j$ et $f(X^j) \not\equiv 0$: nous divisons les pas $\Delta\lambda_i^j$, i = 1,2 par 2 et nous effectuons une itération supplémentaire.

Etape accélération

Dans ce cas, nous cherchons à améliorer de nouveau la valeur de la fonction, c'est-à-dire à la diminuer dans la direction $\overrightarrow{\lambda^j X^j}$ en calculant $f(X'^j)$ pour

$$X'^j = X^j + (X^j - \lambda^j) \qquad (4.2.2)$$

soit :

$$\begin{cases} x'^j_1 = 2\,x_1^j - \lambda_R \\ x'^j_2 = 2\,x_2^j - \lambda_I \end{cases} \qquad (4.2.3)$$

Deux cas se présentent :

a) $f(X'^j) \geqslant f(X^j)$: nous prendrons alors $\lambda^{j+1} = X^j$.

b) $f(X'^j) < f(X^j)$: et nous continuons de chercher dans la direction $\overrightarrow{\lambda^j X^j}$ en prenant :

$$X''^j = X'^j + (X^j - \lambda^j) \qquad (4.2.4)$$

et calculant $f(X''^j)$. Nous aurons alors encore deux cas, etc...

$X^{(p)j}$ sera donné par :

$$X^{(p)j} = X^{(p-1)j} + (X^j - \lambda^j) \qquad (4.2.5)$$

et dès que nous trouverons $f(X^{(p)j}) \geqslant f(X^{(p-1)j})$, nous prendrons $\lambda^{j+1} = X^{(p-1)j}$.

A (4.2.5) nous pouvons préférer la relation :

$$\overrightarrow{\lambda^j X^{(p)j}} = (p+1)\overrightarrow{\lambda^j X^j} \qquad (4.2.6)$$

Nota

Cette accélération est plus"douce" que celle prévue dans |36|, qui remplacerait (4.2.6) par :

$$\overrightarrow{\lambda^j X^{(p)j}} = 2^{p+1}.\overrightarrow{\lambda^j X^j} \qquad (4.2.7)$$

ce qui pourrait nous faire sortir du domaine d'unimodalité de f. Nous préférons choisir un domaine de recherche aussi large que possible au départ et ne pas avoir trop à "accélérer". Nous donnerons donc un choix possible pour $\Delta\lambda_R^0$ et $\Delta\lambda_I^0$.

Détermination des $\Delta\lambda_i^j$ après une étape accélération

Posons :

$$\Delta X_i^j = X_i^j - \lambda_i^j \qquad i = 1,2 \qquad (4.2.8)$$

Connaissant ΔX_i^{j-1} et ΔX_i^j, nous calculons à l'étape j :

$$P_i^j = \Delta X_i^{j-1} . \Delta X_i^j \qquad (4.2.9)$$

Trois cas peuvent se rencontrer :

a) $P_i^j < 0$: qui traduit le fait que d'une itération à l'autre, la variation de λ_i n'est pas monotone, et nous prenons :

$$\Delta\lambda_i^{j+1} = \Delta\lambda_i^j/2 \qquad (4.2.10)$$

b) $P_i^j = 0$: qui traduit le fait qu'à l'une des deux itérations successives la variation de λ_i a été nulle. Nous prenons :

$$\Delta\lambda_i^{j+1} = \Delta\lambda_i^j \qquad (4.2.11)$$

c) $P_i^j > 0$: qui traduit le fait que la variation de λ_i est monotone. Nous gardons :

$$\Delta \lambda_i^{j+1} = \Delta\lambda_i^j \qquad (4.2.12)$$

Tests d'arrêt utilisés :

- Arrêt sur la valeur de la fonction : nous arrêtons les itérations lorsque le nombre de chiffres significatifs de $f(\lambda)$ est inférieur à une valeur que nous allons déterminer.

- Arrêt sur le nombre d'itérations : c'est un arrêt de prudence en cas de valeur propre loin du domaine de recherche et de variation de la fonction inférieure aux erreurs numériques sur $f(\lambda)$. Nous verrons comment nous limitons ces effets.

Cas où $\lambda \in \mathbb{R}$ et A matrice réelle

Le principe reste le même, mais nous réduisons considérablement le volume de calcul par la stratégie suivante.

Connaissant $\Delta\lambda^j$, λ^j et $f(\lambda^j)$, nous choissons $X^j = \lambda^j + \Delta\lambda^j$ et nous calculons $f(X^j)$.

Examinons les cas possibles :

1) j = 1 (1ère itération)

a) $f(X^j) \geq f(\lambda^j)$

Nous prenons $X^j = \lambda^j - \Delta\lambda^j$ (4.2.13)

b) $f(X^j) < f(\lambda^j)$

Nous effectuons l'étape accélération définie plus haut avec (4.2.2) jusqu'à ce que nous trouvions $X^{(p)j}$ tel que $f(X^{(p)j}) \geq f(X^{(p-1)j})$; alors nous choisissons :

$$\lambda^{j+1} = X^{(p-1)j}$$

soit $$\lambda^{j+1} = \lambda^j + p.\Delta\lambda \qquad (4.2.14)$$

où $\Delta\lambda = \Delta\lambda^1$ (pas initial)

et $$f(\lambda^{j+1}) = f(X^{(p-1)j}) \qquad (4.2.15)$$

ainsi que $$\Delta\lambda^2 = \Delta\lambda^1$$

2) j > 1 : nous avons alors déjà borné l'extremum (noté λ^e).

a) $\underline{f(X^j) \geq f(\lambda^j)}$ et nous appliquons (4.2.13).

b) $\underline{f(X^j) < f(\lambda^j)}$

Nous savons que $\lambda^e \in [\lambda^j - 2\Delta\lambda^j, \lambda^j + 2\Delta\lambda^j]$; nous prenons donc :

$$\left.\begin{aligned} \lambda^{j+1} &= X^j,\ f(\lambda^{j+1}) = f(X^j), \\ \Delta\lambda^{j+1} &= \Delta\lambda^j/2 \end{aligned}\right| \qquad (4.2.16)$$

Cas où $X^j = \lambda^j - \Delta\lambda^j$

Recommençons les cas possibles :

1) $\underline{j = 1}$

a) $\underline{f(X^j) \geq f(\lambda^j)}$

Cette fois-ci, nous choisissons :

$$\left.\begin{aligned} \lambda^{j+1} &= \lambda^j,\ f(\lambda^{j+1}) = f(\lambda^j) \\ \Delta\lambda^{j+1} &= \Delta\lambda^j/2 \end{aligned}\right| \qquad (4.2.17)$$

b) $\underline{f(X^j) < f(\lambda^j)}$

Nous effectuons l'étape accélération.(4.2.14) est remplacé par :

$$\lambda^{j+1} = \lambda^j - p.\Delta\lambda \qquad (4.2.18)$$

(4.2.15) est inchangé.

1) $\underline{j > 1}$

a) $\underline{f(X^j) \geq f(\lambda^j)}$: nous appliquons (4.2.17).

b) $\underline{f(X^j) < f(\lambda^j)}$: nous appliquons (4.2.16).

Nous vérifions que l'ensemble des solutions possibles de λ^j après j itérations tend, pour $j \to +\infty$, vers un ensemble dense dans notre domaine de recherche initial.

De plus, si $\Delta\lambda$ est une bonne estimation de l'erreur, le nombre d'opérations effectuées par cet algorithme est minimal : (1 ou 2 calculs de f dès que $j > 1$).

IV.2.4 - Convergence de l'algorithme

Nous examinons dans ce paragraphe seulement la convergence théorique (erreurs numériques nulles). Nous nous ramenons dans les hypothèses de la démonstration donnée dans |36| p. 43.

Considérons le polynôme caractéristique :

$$P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \ldots - p_n \qquad (4.2.19)$$

déjà donné en (3.2.1).

Par définition, nous avons :

$$P(\lambda) = \det(A - \lambda I) \qquad (4.2.20)$$

On peut donc écrire :

$$f(\lambda) = |\det(A - \lambda I)|$$
$$= \text{Réel}^2(P(\lambda)) + \text{Im}^2(P(\lambda)) \qquad (4.2.21)$$

Nous pouvons alors montrer que $f(\lambda)$ est une forme quadratique semi-définie positive dont les 0 sont en nombre fini (ils correspondent aux valeurs propres de A).

Une telle fonction est strictement convexe au voisinage de ses 0 (donc au voisinage des valeurs propres).

Si nous considérons $f(\lambda_R, \lambda_I)$. C'est une fonction dérivable et de dérivées partielles continues. Il suffirait pour cela de développer (4.2.21).

Enfin, il existe $p \in \mathbb{R}$ et $c \in \mathbb{C}$ tels que le disque D(c,p) de centre c et de rayon p contienne toutes les valeurs propres λ_i, racines de $P(\lambda)$. Il est clair alors que :

$$|f(\lambda)| \to +\infty \text{ quand } |\lambda| \to +\infty$$

Toutes les conditions suffisantes de la démonstration de la convergence |36| p. 43 sont réunies.

IV.2.5 - Détection de la singularité numérique par la méthode de permutation

Nous cherchons le zéro numérique de la fonction $f(\lambda) = |\det(A-\lambda I)|^2$. Nous avons vu qu'il nous suffisait de trois résultats numériques représentatifs des résultats numériques possibles pour obtenir une estimation correcte du rapport signal/bruit de nos résultats.

Nous utiliserons ici, en attendant de trouver une méthode plus stable, la méthode de Gauss avec recherche du pivot maximal par colonne pour calculer le déterminant de la matrice $(A-\lambda I)$.

Il a été montré |38| que pour tester la validité des résultats obtenus par cette méthode, il était suffisant de permuter les colonnes de la matrice dont nous cherchons le déterminant. C'est ce que nous ferons ici en effectuant trois calculs sur trois matrices B_1, B_2, B_3 déduites de la matrice $(A-\lambda I)$ par permutation des colonnes.

B_1 sera la matrice $(A-\lambda I)$ non permutée,

B_2 sera déduite de $(A-\lambda I)$ en permutant toutes les colonnes paires avec les colonnes impaires,

B_3 sera déduite de $(A-\lambda I)$ en permutant les colonnes i avec les colonnes n-i+1 pour $i \leqslant n/2$.

Les calculs étant effectués dans un ordre différent, la propagation des erreurs numériques va être différente. D'autre part, les résultats numériques obtenus seront bien distribués autour de la valeur exacte du déterminant.

Analysons maintenant le comportement des résultats numériques de $f(\lambda)$. Nous appelons R le résultat sur le déterminant et R' le résultat sur $f(\lambda)$. Nous possédons trois valeurs R' :

$$\left.\begin{aligned} R'_1 &= S' + E'_1 \\ R'_2 &= S' + E'_2 \\ R'_3 &= S' + E'_3 \end{aligned}\right\} \quad (4.2.6) \text{ avec } S' = f(\lambda) = |\det(A-\lambda I)|^2$$

E'_1, E'_2, E'_3 seront des erreurs numériques issues du calcul de $f(\lambda)$ et R'_1, R'_2 et R'_3 les résultats.

Mais nous voyons ici que pour $f(\lambda) = 0$, $S' = 0$, et néanmoins $R'_k \geqslant 0$ pour $k = 1,2,3$. C'est donc que les valeurs E'_k, $k = 1,2,3$ ne sont plus distribuées comme une variable Gaussienne centrée (réf. II.3.1).

Si R_k est le résultat de $\det(A-\lambda I)$, $k = 1,2,3$, nous écrirons encore :

$$R_k = S + E_k \qquad (4.2.27)$$

où E_k est l'erreur numérique faite sur le calcul théorique S de $\det(A - \lambda I)$.

Considérons les variables aléatoires E'_k et E_k ; leurs influences sur les résultats peuvent être caractérisées par leurs moyennes $\bar{E}'$ et $\bar{E}$, et leurs écarts-types δ' et δ.

E_k est bien une Gaussienne centrée |37| (que ce soit dans $\mathbb{R}$ ou dans $\mathbb{C}$) (donc $\bar{E} = 0$).

Le rapport signal/bruit sur le calcul de $\det(A-\lambda I)$ peut être estimé par : $\frac{|\bar{R}|}{\delta}$ (réf. II.3.1).

Comment alors peut-on estimer le rapport signal/bruit sur le calcul de $f(\lambda)$?

Si $S = 0$, alors $S' = 0$ et E'_k peut être approximée par :

$$E'_k = |E_k|^2 \tag{4.2.28}$$

(c'est-à-dire négliger l'erreur due au calcul de $|E_k|^2$).

Nous allons essayer de déduire $\bar{E}'$ et δ' à partir de δ.

Pour simplifier l'exposé, limitons-nous au cas où $\lambda \in \mathbb{R}$ et A matrice réelle, donc au cas où les valeurs E_k sont aussi réelles.

$$\text{Alors } E'_k = E_k^2 \tag{4.2.29}$$

Nous noterons $ESP(E_k)$ l'espérance de la variable aléatoire E_k (distribuée comme une Gaussienne d'écart-type δ) (réf. II.3.1), c'est-à-dire comme la fonction g(x), telle que :

$$g(x) = \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{x^2}{2\delta^2}} \tag{4.2.30}$$

Nous noterons $\bar{E}$ ou $\bar{E}'$ la moyenne de E_k ou E'_k.

Nous avons :

$$\bar{E}' = ESP(E'_k) = ESP(E_k^2) = \delta^2 \tag{4.2.31}$$

(propriété d'une Gaussienne centrée).

Le calcul de l'écart-type δ' de E'_k est donné par :

$$\begin{aligned} \delta'^2 &= \text{ESP}\left((\bar{E}'_k - E'_k)^2\right) \\ &= \text{ESP}\left(E'^2_k\right) - \bar{E}'^2 \\ &= \text{ESP}\left(E^4_k\right) - \delta^4 \end{aligned} \qquad (4.2.32)$$

Les différents moments d'ordre k d'une variable Gaussienne centrée peuvent être identifiés au kième coefficient du développement limité de $\phi(t)$.

où $\phi(t)$ est la fonction caractéristique de g(x), c'est-à-dire :

$$\phi(t) = e^{-\frac{\delta^2 t^2}{2}} \qquad (4.2.33)$$

donc identifié à :

$$i^k \, \phi^{(k)}(0) \qquad (4.2.34)$$

où $i = \sqrt{-1}$.

Le calcul de $i^4 \, \phi^{(4)}(0)$ donne :

$$\text{ESP}(E^4_k) = 3\,\delta^4 \qquad (4.2.35)$$

soit : $\delta'^2 = 3\,\delta^4 - \delta^4$,

ou : $$\delta' = \delta^2 \sqrt{2} \qquad (4.2.36)$$

Alors $\frac{|\bar{R}'|}{\delta}$ est minimisé par $\frac{|\bar{E}'|}{\delta'} = \frac{1}{\sqrt{2}}$ (4.2.37)

Cette valeur résiduelle montre bien les précautions à prendre pour un test direct sur les résultats de $f(\lambda)$. Nous noterons $|\bar{R}'|\text{Min}$ la valeur résiduelle de $|\bar{R}'|$.

Si $S \neq 0$,

$$S = P(\lambda) \neq 0$$

Nous écrirons :

$$R_k = P(\lambda) + E_k \qquad (4.2.38)$$

ou encore $$\frac{|\bar{R}|}{\delta} = q \neq 0 \qquad (4.2.39)$$

soit : $$|\bar{R}| = q \cdot \delta = P(\lambda) \qquad (4.2.40)$$

Si nous développons, nous obtenons :

$$\bar{R}' = ESP((R_k)^2) = ESP[(P(\lambda) + E_k)^2]$$

$$= ESP[P(\lambda)^2 + 2P(\lambda)E_k + E_k^2]$$

$$= P(\lambda)^2 + 2P(\lambda)\, ESP(E_k) + ESP(E_k^2)$$

$$= q^2 . \delta^2 + 2q.\delta . \times 0 + \delta^2$$

$$= (q^2 + 1)\, \delta^2 \qquad (4.2.41)$$

$$\delta'^2 = ESP[(\bar{R}' - R_k')^2]$$

$$= ESP[R_k'^2] - \bar{R}'^2 = ESP[R_k^4] - \bar{R}'^2$$

$$= ESP[P(\lambda)^4 + 4P(\lambda)^3 E_k + 6P(\lambda)^2 E_k^2 + 4P(\lambda) E_k^3 + E_k^4] - \bar{R}'^2$$

Or $$\phi'(0) = \phi'''(0) = 0$$

d'où :

$$\delta'^2 = P(\lambda)^4 + 6P(\lambda)^2\, ESP[E_k^2] + ESP[E_k^4] - \bar{R}'^2$$

Soit :

$$\delta'^2 = P(\lambda)^4 + 6P(\lambda)^2\, \delta^2 + 3\, \delta^4 - \bar{R}'^2$$

$$= q^4 . \delta^4 + 6q^2\, \delta^4 + 3\, \delta^4 - (q^2 + 1)^2\, \delta^4$$

$$= 2(2q^2 + 1)\, \delta^4 \qquad (4.2.42)$$

Soit un rapport :

$$\frac{|\bar{R}'|}{\delta'} = \frac{(q^2+1)}{\sqrt{2q^2+1}\ \sqrt{2}} \qquad (4.2.43)$$

donc, pour un rapport S/B = 1 : q = 1, nous obtiendrons :

$$S'/B' = \sqrt{\frac{2}{3}}$$

IV.2.6 - Mise en oeuvre

1) Comparaison numérique de $f(X^j)$ avec $f(\lambda^j)$

Là encore, des précautions sont nécessaires si nous ne voulons pas que le test $f(X^j) > f(\lambda^j)$ donne un résultat faux dû aux erreurs numériques. Il est évident que pour $f(\lambda^j)$ tendant vers zéro, le risque augmente.

Si nous écrivons :

$$R'_k(X^j) = f(X^j) + E'_k(X^j) \qquad (4.2.44)$$

$$R'_k(\lambda^j) = f(\lambda^j) + E'_k(\lambda^j) \qquad (4.2.45)$$

et si nous supposons que $f(\lambda^j) - f(X^j) > 0$, quel est le risque que $R'_k(\lambda^j) - R'_k(X^j) < 0$?

$$R'_k(\lambda^j) - R'_k(X^j) = f(\lambda^j) - f(X^j) - (E'_k(X^j) - E'_k(\lambda^j)) \qquad (4.2.46)$$

$$R'_k(\lambda^j) - R'_k(X^j) < 0 \iff \boxed{\frac{f(\lambda^j)-f(X^j)}{E'_k(X^j) - E'_k(\lambda^j)} < 1} \qquad (4.2.47)$$

Restreignons-nous encore au cas où $\lambda \in \mathbb{R}$ et A matrice réelle, λ valeur propre théorique.

a) $\lambda \in [\lambda^j - \Delta\lambda^j, \lambda^j + \Delta\lambda^j]$

Soit par exemple : $\lambda \in [\lambda^j - \Delta\lambda^j, \lambda^j - \Delta\lambda^j/2]$ et $f(\lambda^j - \Delta\lambda^j) < f(\lambda^j)$.

Alors, en théorie, $\lambda^{j+1} = \lambda^j - \Delta\lambda^j$.

Soit alors $R'_k(\lambda^j - \Delta\lambda^j) > R'_k(\lambda^j)$. A l'étape suivante, nous calculerons :

$$f(\lambda^{j+1} - \Delta\lambda^{j+1}) \quad \text{soit} \quad f(\lambda^j - \Delta\lambda^j/2)$$

qui sera comparé au résultat de $f(\lambda^{j+1})$, soit $f(\lambda^j)$.

De par la convexité de f au voisinage de λ(réf. IV.2.4) : $\forall\ \theta \in [0,1]$ nous avons :

$$(1-\theta)\ f(\lambda) + \theta\, f(\lambda^j) \geqslant f((1-\theta)\lambda + \theta\lambda^j) \qquad (4.2.48)$$

Ceci donne, en choissant θ tel que $(1-\theta)\lambda + \theta\lambda^j = \lambda^j - \Delta\lambda^j/2$:

$$\theta\, f(\lambda^j) \geqslant f(\lambda^j - \Delta\lambda^j/2) \qquad (4.2.49)$$

Puisque $\lambda \in [\lambda^j - \Delta\lambda^j,\ \lambda^j - \Delta\lambda^j/2]$ on a $\theta \leqslant 1/2$, d'où $\frac{1}{\theta} \geqslant 2$.

Nous obtenons donc : $f(\lambda^j) \geqslant 2\, f(\lambda^j - \Delta\lambda^j/2)$, soit :

$$f(\lambda^j) - f(\lambda^j - \Delta\lambda^j/2) \geqslant f(\lambda^j - \Delta\lambda^j/2) \qquad (4.2.50)$$

Une nouvelle erreur numérique au test $f(X^{j+1}) > f(\lambda^{j+1})$ entraînerait alors, d'après (4.2.27) et (4.2.50) :

$$E'_k(\lambda^j - \Delta\lambda^j/2) - E'_k(\lambda^j) \geqslant f(\lambda^j - \Delta\lambda^j/2) \qquad (4.2.51)$$

Or, un test d'arrêt sur la précision de $f(\lambda^j - \Delta\lambda^j/2)$ peut empêcher cela. Il suffit de tester si $\frac{|\bar{R}'|}{\delta'} < 2$ (si nous faisons l'hypothèse que $E'_k(\lambda^j)$ est du même ordre de grandeur que $E'_k(\lambda^j - \Delta\lambda^j/2)$). Par ailleurs, une comparaison sur les résultats moyens $\bar{R}'(X^j)$ et $\bar{R}'(\lambda^j)$ diminue le risque d'erreur.

Nous montrons de même, en traitant tous les cas possibles pour $\lambda \in [\lambda^j - \Delta\lambda^j,\ \lambda^j + \Delta\lambda^j]$ que si le test sur $\frac{|\bar{R}'|}{\delta'}$ est convenablement choisi, une erreur numérique sur un test n'entraîne pas une divergence de la solution.

b) $\underline{\lambda \in [\lambda^j - \Delta\lambda^j,\ \lambda^j + \Delta\lambda^j]}$

Supposons encore, pour fixer les idées, que :

$$\lambda < \lambda^j$$

et utilisons encore l'équation de convexité (4.2.48).

Pour θ convenablement choisi, cette équation ramène encore à :

$$f(\lambda^j) \geqslant \frac{1}{\theta} f(\lambda^j - \Delta\lambda^j) \qquad (4.2.52)$$

soit :

$$f(\lambda^j) - f(\lambda^j - \Delta\lambda^j) \geqslant (\frac{1}{\theta} - 1)\ f(\lambda^j - \Delta\lambda^j)$$

ou encore :

$$f(\lambda^j) - f(\lambda^j - \Delta\lambda^j) \geqslant (\frac{1-\theta}{\theta})\ f(\lambda^j - \Delta\lambda^j) \qquad (4.2.53)$$

Si, pour simplifier, nous prenons le cas extrême où :

$$E_k'(\lambda^j) = - E_k'(\lambda^j - \Delta\lambda^j)$$

une erreur du choix de λ^{j+1} peut arriver si :

$$E_k'(\lambda^j - \Delta\lambda^j) \geqslant \frac{1}{2} (\frac{1-\theta}{\theta})\ f(\lambda^j - \Delta\lambda^j) \qquad (4.2.54)$$

ou encore peut être évitée si nous effectuons un test d'arrêt sur $f(\lambda^j - \Delta\lambda^j)$ dès que $\frac{|\bar{R}'|}{\delta'} < 2\ \frac{\theta}{1-\theta}$.

θ est fonction de l'éloignement de λ par rapport à λ^j et $\lambda^j - \Delta\lambda^j$ et vérifie l'équation :

$$\theta = 1 - \frac{\Delta\lambda^j}{\lambda^j - \lambda}$$

Nous aboutissons ainsi à une notion d'éloignement admissible pour λ conditionné par le test d'arrêt $\frac{|\bar{R}'|}{\delta'} < q'$.

Conséquence :

Il faut déterminer q' et choisir $\Delta\lambda^0$ optimal. Tout ceci étant tempéré par le fait que nous comparerons $\bar{R}'(X^j)$ avec $\bar{R}'(\lambda^j)$ et que la stricte convexité de f permet en fait une distance de λ plus grande. Mais, par ailleurs, le cas de la variable complexe rend plus imprévisibles ces comportements numériques, même si la convexité joue encore un rôle important.

2) Choix de q'

Nous avons vu avec (4.2.43) que q' était déterminé par $q = \frac{|\bar{R}|}{\delta}$

Pour $q > 2$, nous pouvons approximer q^2+1 par q^2. Alors :

$$\frac{|\bar{R}'|_{Min}}{\delta'} = \frac{q^2+1}{\sqrt{2q^2+1}\sqrt{2}} \simeq \frac{q}{2} \qquad (4.2.55)$$

Une sécurité maximale pour les comparaisons entre $f(X^j)$ et $f(\lambda^j)$ nous incite à prendre q' le plus grand possible. Par ailleurs, l'objectif désigné étant d'obtenir une précision sur λ maximale, donc une précision minimale de $\det(A-\lambda I)$, il faudrait en toute rigueur choisir q minimum, donc au moins inférieur à 10 (si nous voulons que le déterminant n'ait plus aucun chiffre significatif).

Si nous voulons être certain que $q' \geqslant 2$ alors que nous disposons seulement de 3 échantillons pour estimer $\frac{|\bar{R}'|}{\delta'}$, nous prendrons par précaution $\frac{|\bar{R}'|_{Min}}{\delta'} \geqslant 2.48*2$ (réf. II.3.1).

Nous avons donc $2.48*2 \leqslant q' \leqslant \frac{10}{2}$. Nous pouvons choisir q' = 5, c'est-à-dire effectuer un test d'arrêt sur f de la forme :

$$\mathrm{Log}_{10} \frac{|\bar{R}'|}{\delta'} < 0.7 \qquad (4.2.56)$$

3) Choix de $\Delta\lambda^0$

Le meilleur choix possible est d'éviter l'étape accélération, soit de prendre $\Delta\lambda^0 = \lambda - \lambda^0$. ($\lambda^0$ étant la valeur propre calculée, λ la valeur théorique).

Si $C(\lambda^0)$ est le nombre de chiffres significatifs (en valeur réelle), $\Delta\lambda^0$ peut être estimé par :

$$\Delta\lambda^0 = \lambda^0 * 10^{-C(\lambda^0)} \qquad (4.2.57)$$

En fait, pour garder un intervalle de confiance suffisant, nous choisirons dans la mesure du possible :

$$\Delta\lambda^0 = \lambda^0 * 10^{-C(\lambda^0)} * 2.48 \qquad (4.2.58)$$

Pour λ complexe, nous aurons :

$$\left.\begin{aligned} \Delta\lambda_R^0 &= \lambda_R^0 * 10^{-C(\lambda_R^0)} * 2.48 \\ \Delta\lambda_I^0 &= \lambda_I^0 * 10^{-C(\lambda_I^0)} * 2.48 \end{aligned}\right| \qquad (4.2.59)$$

Néanmoins, un recouvrement des pavés dans $\mathbb{R}$ ou $\mathbb{C}$ ainsi définis pourra entraîner parfois une convergence vers d'autres valeurs propres. Il est donc nécessaire que ces valeurs propres aient été suffisamment séparées.

Si PRÉ est le nombre de chiffres significatifs demandés, nous arrêterons la convergence pour :

$$\mathrm{Log}_{10}\left|\frac{\lambda^j}{\Delta\lambda^j}\right| < \text{PRÉ}$$

IV.2.7 - Notice d'utilisation des programmes AOPTIM et COPTIM

Objet

Amélioration de la valeur propre.

Appel

```
CALL AOPTIM(VR,EVR,A,C,IA,N,PRE)
```

Arguments d'entrée

VR Valeur propre initiale.

EVR Estimation de l'erreur sur la valeur propre.

A Tableau bidimensionné dont la première dimension est IA et la seconde au moins égale à N. A contient la matrice dont nous cherchons la valeur propre.

C Tableau de travail de dimensions égales à A.

IA Première dimension des tableaux A et C.

N Ordre de la matrice A.

PRE Nombre de chiffres significatifs demandés (valeur réelle).

Arguments de sortie

VR Valeur propre améliorée.

EVR Estimation de l'erreur sur la valeur propre améliorée.

Le programme COPTIM possède les mêmes arguments que le programme AOPTIM, mais VR, EVR, A et C sont COMPLEXES.

Le programme AOPTIM utilise le sous-programme DETVP testant la validité de la fonction $f(\lambda)$ et donnant sa valeur. Il utilise lui-même le programme DETG, DETG étant un sous-programme de calcul du déterminant par la méthode de Gauss avec recherche du pivot maximal par colonne.

Le sous-programme WXTEST testant la validité d'un résultat est aussi utilisé.

COPTIM utilise respectivement CDETVP et CDETG.

IV.2.8 - Programmes FORTRAN COPTIM et AOPTIM.

```
      SUBROUTINE COPTIM(VAL,DVAL,A,B,IA,NA)                                  00000250
      DIMENSION K(2),PASK(2),DELK(6),KBON(6),VALKBA(6),A(IA,1),B(IA,1)       00000260
     1 ,DIFK(6),DIFKBA(6),VALK(6),KR(6),VARA(6),KV(6,6),NOM(6)               00000270
      COMPLEX VAL,DVAL                                                       00000280
      LOGICAL RETOUR                                                         00000290
      REAL K,KBON,KR,KV                                                      00000300
      K(1)=REAL(VAL)                                                         00000310
      K(2)=AIMAG(VAL)                                                        00000320
      PASK(1)=REAL(DVAL)                                                     00000330
      PASK(2)=AIMAG(DVAL)                                                    00000340
      NINC=2                                                                 00000350
      NBITER=100                                                             00000360
      FACTPB=1.                                                              00000370
      FACTPA=2.                                                              00000380
      DO 1 I=1,NINC                                                          00000390
      NOM(I)=6H                                                              00000400
      VALKBA(I)=K(I)                                                         00000410
    1 KBON(I)=K(I)                                                           00000420
      IS=0                                                                   00000430
      N=0                                                                    00000440
      CALL CDETVP(K,EQMIN,A,B,IA,NA,RETOUR)                                  00000450
      IF(RETOUR)GOTO 29                                                      00000460
      WRITE(6,201)N,EQMIN                                                    00000470
      EQBON=EQMIN                                                            00000480
    2 DO 3 I=1,NINC                                                          00000490
      KR(I)=K(I)                                                             00000500
      IF(ABS(KR(I)).LT.1.)KR(I)=1.                                           00000510
    3 CONTINUE                                                               00000520
    5 NLIM1=3*NINC                                                           00000530
      DO 6 I=1,NLIM1                                                         00000540
      I1=I-((I-1)/3)*3                                                       00000550
      I2=(I-1)/3+1                                                           00000560
    6 KV(I2,I1)=K(I2)+(FLOAT(I1)-2.)*PASK(I2)*KR(I2)                         00000570
      NLIM2=3**NINC                                                          00000580
      DO 9 I=1,NLIM2                                                         00000590
      DO 7 J=1,NINC                                                          00000600
      L=MOD((I-1)/(3**(NINC-J)),3)+1                                         00000610
    7 K(J)=KV(J,L)                                                           00000620
      CALL CDETVP(K,EQ,A,B,IA,NA,RETOUR)                                     00000630
      IF(RETOUR)GOTO 28                                                      00000640
      IF(EQ.GE.EQBON)GO TO 9                                                 00000650
      EQBON=EQ                                                               00000660
      DO 8 II=1,NINC                                                         00000670
      KBON(II)=K(II)                                                         00000680
    8 CONTINUE                                                               00000690
    9 CONTINUE                                                               00000700
   10 IF(EQBON.GE.EQMIN) GO TO 19                                            00000710
      EQR=EQBON                                                              00000720
      IF(IS.EQ.1) GO TO 15                                                   00000730
      DO 11 I=1,NINC                                                         00000740
   11 DELK(I)= KBON(I)-VALKBA(I)                                             00000750
   12 DO 13 I=1,NINC                                                         00000760
   13    K(I)=KBON(I)+DELK(I)                                                00000770
      CALL CDETVP(K,EQ,A,B,IA,NA,RETOUR)                                     00000780
      IF(RETOUR)GOTO 28                                                      00000790
      IF(EQ.GE.EQR)    GO TO 15                                              00000800
      EQR=EQ                                                                 00000810
```

```
                  DO 14 I=1,NINC                                                00000820
                  DELK(I)=2.*DELK(I)                                            00000830
60                14 KBON(I)=K(I)                                               00000840
                  GO TO 12                                                      00000850
                  15 DO 16 I=1,NINC                                             00000860
                  DIFKBA(I)=DIFK(I)                                             00000870
                  16 DIFK(I)=KBON(I)-VALKBA(I)                                  00000880
65                IS=0                                                          00000890
                  EQMIN=EQR                                                     00000900
                  EQBON=EQR                                                     00000910
                  N=N+1                                                         00000920
                  IF(N.GT.NBITER) GO TO 27                                      00000930
70                GOTO 21                                                       00000940
                  19 DO 20 I=1,NINC                                             00000950
                  20 PASK(I)=PASK(I)/FACTPA                                     00000960
                  IS=1                                                          00000970
                  GO TO 25                                                      00000980
75                21 IF(N.LT.2) GO TO 25                                        00000990
                  DO 24 I=1,NINC                                                00001000
                  IF(DIFK(I)*DIFKBA(I)) 22,24,23                                00001010
                  22 PASK(I)=PASK(I)/FACTPA                                     00001020
                  GO TO 24                                                      00001030
80                23 PASK(I)=PASK(I)*FACTPB                                     00001040
                  24 CONTINUE                                                   00001050
                  25 DO 26 I=1,NINC                                             00001060
                  VALKBA(I)=KBON(I)                                             00001070
                  26 K(I)=KBON(I)                                               00001080
85                GO TO 2                                                       00001090
               29   WRITE(6,200)                                               00001100
               28   CONTINUE                                                   00001110
                    VAL=CMPLX(K(1),K(2))                                       00001120
                    DVAL=CMPLX(PASK(1),PASK(2))                                00001125
90                  RETURN                                                     00001130
               27   WRITE(6,202)NBITER                                         00001140
                    GOTO 28                                                    00001150
              200   FORMAT(1H,1X,27HPAS D'AMELIORATION POSSIBLE    )           00001160
                201 FORMAT(/,10X,3HEQ(,I4,4H) = ,E15.8,//)                     00001170
95              202 FORMAT(1H ,/////30X,34HPRECISION DEMANDEE NON OBTENUE EN ,I4,  00001180
                   112H ITERATIONS.,/)                                         00001190
                    END                                                        00001200
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 COPTIM

VARIABLES	SN	TYPE	RELOCATION						
0	A	REAL	ARRAY	F.P.	0	B	REAL	ARRAY	F.P.
375	DELK	REAL	ARRAY		417	DIFK	REAL	ARRAY	
425	DIFKBA	REAL	ARRAY		0	DVAL	COMPLEX		F.P.
366	EQ	REAL			357	EQBON	REAL		
356	EQMIN	REAL			370	EQR	REAL		
352	FACTPA	REAL			351	FACTPB	REAL		

```
      SUBROUTINE CDETVP(K,EQMIN,A,B,IA,N,RETOUR)                        00001210
      DIMENSION K(2),A(IA,1),B(IA,1),D(3)                               00001220
      COMMON/WXTST/CMIN,CMAX                                            00001230
      COMPLEX A,B,VAL,DELTA                                             00001240
      LOGICAL RETOUR                                                    00001250
      REAL K                                                            00001260
      CMIN=0.7                                                          00001270
      CMAX=15.                                                          00001280
      VAL=CMPLX(K(1),K(2))                                              00001290
      DO 1 I=1,N                                                        00001300
      DO 2 J=1,N                                                        00001310
    2 B(I,J)=A(I,J)                                                     00001320
    1 B(I,I)=B(I,I)-VAL                                                 00001330
      CALL CDETG(B,IA,N,DELTA)                                          00001340
      D1=REAL(DELTA)                                                    00001350
      D2=AIMAG(DELTA)                                                   00001360
      D(1)=D1*D1+D2*D2                                                  00001370
      DO 6 I=2,N,2                                                      00001380
      DO 6 J=1,N                                                        00001390
      B(J,I-1)=A(J,I)                                                   00001400
    6 B(J,I)=A(J,I-1)                                                   00001410
      DO 5 I=2,N,2                                                      00001420
      B(I-1,I)=B(I-1,I)-VAL                                             00001430
    5 B(I,I-1)=B(I,I-1)-VAL                                             00001440
      CALLCDETG(B,IA,N,DELTA)                                           00001441
      D1=REAL(DELTA)                                                    00001442
      D2=AIMAG(DELTA)                                                   00001450
      D(2)=D1*D1+D2*D2                                                  00001460
      DO 3 I=1,N                                                        00001540
      L=N-I+1                                                           00001550
      DO 4 J=1,N                                                        00001560
    4 B(J,L)=A(J,I)                                                     00001570
    3 B(I,L)=B(I,L)-VAL                                                 00001580
      CALL CDETG(B,IA,N,DELTA)                                          00001581
      D1=REAL(DELTA)                                                    00001582
      D2=AIMAG(DELTA)                                                   00001583
      D(3)=D1*D1+D2*D2                                                  00001584
      CALL WXTEST(D,3,CD,EQMIN,RETOUR)                                  00001585
      WRITE(6,*)D(1),D(2),D(3),VAL                                      00001590
      RETURN                                                            00001600
      END                                                               00001610
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 CDETVP

VARIABLES	SN	TYPE		RELOCATION					
0	A	COMPLEX	ARRAY	F.P.	0	B	COMPLEX	ARRAY	F.P.
253	CD	REAL			1	CMAX	REAL		WXTST
0	CMIN	REAL		WXTST	254	D	REAL	ARRAY	
244	DELTA	COMPLEX			250	D1	REAL		
251	D2	REAL			0	EQMIN	REAL		F.P.

```
      1              SUBROUTINE CDETG(A,NA1,N,DELTA)                          00001900
                     DIMENSION A(NA1,1)                                       00001910
                     COMPLEX A,DELTA,AUX                                      00001920
                     DELTA=CMPLX(1.,0.)                                       00001930
      5              DO 6 I=1,N                                               00001940
                     IP1=I+1                                                  00001950
                     AMAX=0.                                                  00001960
                     L=I                                                      00001970
                     DO 1 K=I,N                                               00001980
     10              IF(AMAX.GE.CABS(A(K,I)))GOTO 1                           00001990
                     L=K                                                      00002000
                     AMAX=CABS(A(K,I))                                        00002010
                1    CONTINUE                                                 00002020
                     DELTA=DELTA*A(L,I)                                       00002030
     15              IF(AMAX.EQ.0..OR.I.EQ.N)RETURN                           00002040
                     IF(L.EQ.I)GOTO 3                                         00002050
                     DO 2 J=I,N                                               00002060
                     AUX=A(I,J)                                               00002070
                     A(I,J)=A(L,J)                                            00002080
     20              A(L,J)=AUX                                               00002090
                2    DELTA=-DELTA                                             00002100
                3    DO 4 J=IP1,N                                             00002110
                4    A(I,J)=A(I,J)/A(I,I)                                     00002120
                     DO 5 K=IP1,N                                             00002130
     25              DO 5 J=IP1,N                                             00002140
                5    A(K,J)=A(K,J)-A(K,I)*A(I,J)                              00002150
                6    CONTINUE                                                 00002160
                     END                                                      00002170
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 CDETG

VARIABLES	SN	TYPE	RELOCATION					
0	A	COMPLEX	ARRAY	F.P.	166	AMAX	REAL	
162	AUX	COMPLEX			0	DELTA	COMPLEX	F.P.
164	I	INTEGER			165	IP1	INTEGER	
171	J	INTEGER			170	K	INTEGER	
167	L	INTEGER			0	N	INTEGER	F.P.
0	NA1	INTEGER		F.P.				

EXTERNALS	TYPE	ARGS
CABS	REAL	1 LIBRARY

INLINE FUNCTIONS	TYPE	ARGS
CMPLX	COMPLEX	2 INTRIN

STATEMENT LABELS

40	1	0	2	75	3
0	4	0	5	0	6

```
      SUBROUTINE AOPTIM(VR,EVR,A,C,IA,N,PRE)                            00000420
      DIMENSION A(IA,1),C(IA,1),D(3)                                    00000430
      LOGICAL RETOUR                                                    00000440
      EPS=10**(-PRE)*VR                                                 00000450
      INC=0                                                             00000460
      JNC=0                                                             00000470
      VAL=VR                                                            00000480
      CALL DETVP(VAL,D(2),A,C,IA,N,RETOUR)                              00000490
      IF(RETOUR)GOTO 29                                                 00000500
1     CONTINUE                                                          00000510
      PRINT *,VR,EVR,D(2),INC                                           00000520
      IF(EVR.LT.EPS)GOTO 28                                             00000530
      INC=INC+1                                                         00000540
4     IF(INC.GT.30)GOTO 30                                              00000550
      JNC=JNC+1                                                         00000560
      IF(JNC.GT.40)GOTO 30                                              00000570
      VAL=VR-EVR                                                        00000580
      CALL DETVP(VAL   ,D(1),A,C,IA,N,RETOUR)                           00000590
      IF(RETOUR)GOTO 28                                                 00000600
      IF(D(1).GE.D(2))GOTO 2                                            00000610
      D(2)=D(1)                                                         00000620
      VR=VAL                                                            00000630
      IF(INC.EQ.1)GOTO 4                                                00000640
      EVR=EVR/2.                                                        00000650
      GOTO 1                                                            00000660
2     CONTINUE                                                          00000670
      VAL=VR+EVR                                                        00000680
      CALL DETVP(VAL   ,D(3),A,C,IA,N,RETOUR)                           00000690
      IF(RETOUR)GOTO 28                                                 00000700
      IF(D(3).GE.D(2))GOTO 3                                            00000710
      D(2)=D(3)                                                         00000720
      VR=VAL                                                            00000730
      IF(INC.EQ.1)GOTO 4                                                00000740
3     EVR=EVR/2.                                                        00000750
      GOTO 1                                                            00000760
29    WRITE(6,100)                                                      00000770
100   FORMAT(1H1,1X,27HPAS D'AMELIORATION POSSIBLE)                     00000780
28     CONTINUE                                                         00000790
      EVR=EVR/2.                                                        00000800
      VR=VAL                                                            00000810
      RETURN                                                            00000820
30     WRITE(6,101)ITER                                                 00000830
101   FORMAT(1H1,1X,21HNON CONVERGENCE APRES,I5,11H ITERATIONS)         00000840
      RETURN                                                            00000850
      END                                                               00000860
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 AOPTIM

```
      SUBROUTINE DETVP(VAL,EQMIN,A,B,IA,N,RETOUR)                       00000870
      DIMENSION A(IA,1),B(IA,1),D(3),MUL(3)                             00000880
      COMMON/WXTST/CMIN,CMAX                                            00000890
      DATA MUL/1.,3.,5./                                                00000895
      CMIN=0.7                                                          00000900
      CMAX=15.                                                          00000910
C                                                                       00000912
C     CREATION DE LA MATRICE                                            00000914
C                                                                       00000916
      DO 1 IS=1,3                                                       00000918
      DO 2 I=1,N                                                        00000920
      L=N-I+1                                                           00000921
      DO 3 J=1,N                                                        00000922
    3 B(L,J)=A(I,J)                                                     00000924
      B(L,I)=B(L,I)-VAL                                                 00000926
      DO 2 J=1,N                                                        00000928
    2 B(L,J)=B(L,J)*MUL(IS)                                             00000930
C                                                                       00000932
C     CALCUL DU DETERMINANT                                             00000934
C                                                                       00000936
      D(IS)=0.                                                          00000938
      DET=1.                                                            00000940
      NM1=N-1                                                           00000942
      DO 4 I=1,NM1                                                      00000944
      IP1=I+1                                                           00000946
      AMAX=0.                                                           00000948
      L=I                                                               00000950
      M=I                                                               00000952
C                                                                       00000954
C     RECHERCHE DU PIVOT MAXIMAL                                        00000956
C                                                                       00000958
      DO 5 J=I,N
      DO 5 K=I,N                                                        00000960
      AUX=ABS(B(K,J))                                                   00000962
      IF(AMAX.GE.AUX)GOTO 5                                             00000964
      L=K                                                               00000966
      M=J
      AMAX=AUX                                                          00000968
    5 CONTINUE                                                          00000970
      IF(L.EQ.I)GOTO 8                                                  00000994
      DO 10 J=I,N                                                       00000996
      AUX=B(I,J)                                                        00000998
       B(I,J)=B(L,J)                                                    00000999
   10 B(L,J)=AUX                                                        00001000
    8 IF(M.EQ.I)GOTO 9                                                  00001002
      DO 11 J=I,N                                                       00001004
      AUX=B(J,I)                                                        00001006
      B(J,I)=B(J,M)                                                     00001008
   11  B(J,M)=AUX                                                       00001010
C                                                                       00001012
C     LE PIVOT EST TROUVE                                               00001014
C                                                                       00001016
    9  DET=DET*(B(I,I)/MUL(IS))                                         00001018
      DO 12 J=IP1,N                                                     00001020
   12 B(I,J)=B(I,J)/B(I,I)                                              00001022
      DO 13 K=IP1,N                                                     00001024
      DO 13 J=IP1,N                                                     00001026
```

```
                13      B(K,J)=B(K,J)-B(K,I)*B(I,J)                                00001028
                4       CONTINUE                                                   00001030
     60                 D(IS)=DET*(B(N,N)/MUL(IS))                                 00001032
                7       CONTINUE                                                   00001034
                1       D(IS)=D(IS)*D(IS)                                          00001036
                C                                                                  00001038
                C       TEST SUR LE DETERMINANT                                    00001040
     65         C                                                                  00001042
                      CALL WXTEST(D,3,CD,EQMIN,RETOUR)                             00001130
                      PRINT *,D(1),D(2),D(3)                                       00001140
                      RETURN                                                       00001150
                      END                                                          00001160
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 DETVP

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	271	AMAX	REAL		
274	AUX	REAL			0	B	REAL	ARRAY	F.P.
275	CD	REAL			1	CMAX	REAL		WXTST
0	CMIN	REAL		WXTST	276	D	REAL	ARRAY	
266	DET	REAL			0	EQMIN	REAL		F.P.
263	I	INTEGER			0	IA	INTEGER		F.P.
270	IP1	INTEGER			262	IS	INTEGER		
265	J	INTEGER			273	K	INTEGER		
264	L	INTEGER			272	M	INTEGER		
301	MUL	INTEGER	ARRAY		0	N	INTEGER		F.P.
267	NM1	INTEGER			0	RETOUR	REAL		F.P.
0	VAL	REAL		F.P.					

FILE NAMES	MODE
OUTPUT	FREE

EXTERNALS	TYPE	ARGS
WXTEST		5

INLINE FUNCTIONS	TYPE	ARGS	
ABS	REAL	1	INTRIN

STATEMENT LABELS

0	1	0	2	0	3			
0	4	103	5	0	7	INACTIVE		
120	8	133	9	0	10			
0	11	0	12	0	13			

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
26	1	IS	10 62	164B	NOT INNER
32	2	I	11 17	22B	NOT INNER
37	3	J	13 14	2B	INSTACK
50	2	J	16 17	2B	INSTACK
67	4	I	24 59	115B	NOT INNER

IV.2.9 - Résultats numériques

Après avoir testé la convergence de nos algorithmes, exemple donné ci-après avec COPTIM, nous avons cherché à améliorer les résultats provenant du programme CHSIQR à l'aide du programme AMONIT utilisant AOPTIM (voir pages suivantes). Nous nous apercevons que nous n'avons quasiment rien amélioré.

Ce fait peut s'expliquer à la fois par le mauvais conditionnement de notre matrice et par l'instabilité de la méthode de Gauss. La précision obtenue sur les valeurs propres est du même ordre que la précision obtenue par la méthode QR.

Nous avons cherché à améliorer les résultats en cherchant un pivot maximal par ligne et par colonne et en utilisant donc la méthode de multiplication au lieu de la méthode de permutation, cette dernière n'étant plus applicable. Nous n'avons pas obtenu plus d'amélioration.

Nous concluons donc que si nous voulons améliorer les valeurs propres à partir du calcul du déterminant, il nous faut un calcul de ce déterminant plus stable.

Nous allons proposer en IV.2.10 une autre méthodologie pour l'amélioration de la valeur propre.

Test du programme COPTIM

```
      PROGRAM TEST1(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)
      DIMENSION C(20,20),A(20,20)
      COMPLEX A,C,VAL,EVR
      N=16
      IA=20
      DO 1 I=1,N
      DO 2 J=1,I
    2 A(I,J)=0.
      DO 3 J=I,N
    3 A(I,J)=N-J+1
      IF(I.EQ.1)GOTO 1
      A(I,I-1)=N-I+1
    1 CONTINUE
      VAL=(.17,.01)
      EVR=(.015,.006)
      CALL COPTIM(VAL,EVR,A,C,IA,N)
      PRINT *,VAL, EVR
      STOP
      END
```

Dans ce test, nous avons une approximation de la valeur propre λ = .17 753... calculée en double précision en II.4.4. Nous partons d'une approximation fictive à laquelle nous avons ajouté une fausse composante complexe. Soit :

$$\lambda_R = .17 \qquad \lambda_I = 0.01$$

Nous avons supposé que l'estimation de l'erreur était connue et égale à :

$$\Delta\lambda_R = 0.015 \quad \text{et} \quad \Delta\lambda_I = .006$$

Après un certain nombre d'itérations, notre programme COPTIM détecte une instabilité du déterminant pour la valeur propre complexe :

$$\lambda_R = .1775 \quad \text{et} \quad \lambda_I = .0000625$$

le pas sur chacune des composantes étant alors :

$$\Delta\lambda_R = .0075 \quad \text{et} \quad \Delta\lambda_I = .0001875$$

Nous pouvons constater qu'il y a effectivement un accroissement de la précision vers la valeur propre exacte la plus proche (résultats appréciables pour la composante complexe).

Néanmoins, nous regrettons un arrêt des itérations un peu précoce, dû à l'instabilité du calcul du déterminant par la méthode de Gauss.

Nous avons alors utilisé le programme AOPTIM avec les résultats du programme CHSIQR (1ère version). Nous nous apercevons que les valeurs propres mal conditionnées pour l'algorithme QR le sont aussi pour le calcul du déterminant de $(A-\lambda I)$ par la méthode de Gauss, et donc que les valeurs propres ne sont pas améliorées.

```
      1              PROGRAM TEST1(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)                00000100
                     DIMENSION C(20,20),A(20,20,3),VP(20,2),CHSIVP(20,2),                00000130
                    1 VECT(20,20,3),ITYPE(20)                                            00000140
                     N=16                                                                00000150
      5              IA=20                                                               00000160
                     DO 1 I=1,N                                                          00000170
                     DO 2 J=1,I                                                          00000180
                  2  A(I,J,1)=0.                                                        00000190
                     DO 3 J=I,N                                                          00000200
     10           3  A(I,J,1)=N-J+1                                                     00000210
                     IF(I.EQ.1)GOTO 1                                                    00000220
                     A(I,I-1,1)=N-I+1                                                    00000230
                  1  CONTINUE                                                          00000240
                     DO 5 I=1,N                                                          00000250
     15              DO 5 J=1,N                                                          00000260
                  5  C(I,J)=A(I,J,1)                                                   00000270
                     IARG=0                                                              00000280
                     CALL CHSIQR(A,N,IA,1,N,VP,CHSIVP,VECT,ITYPE,IARG)                   00000290
                     PRE=10.                                                             00000300
     20              CALL AMONIT(VP,CHSIVP,C,A,IA,N,PRE)                                 00000310
                     PRINT 7                                                             00000320
                  7  FORMAT(1H1,5X,2HNO,5X,13HPARTIE REELLE,7X,11HCHIFF.SIGN.,10X,       00000330
                    1   17HPARTIE IMAGINAIRE,3X,11HCHIFF.SIGN.)                        00000340
                     DO 4 I=1,N                                                          00000350
     25              N1=CHSIVP(I,1)                                                      00000360
                     N2=CHSIVP(I,2)                                                      00000370
                  4  PRINT 6,I,VP(I,1),N1,VP(I,2),N2                                   00000380
                  6  FORMAT(/2X,I5,3X,E21.14,7X,I2,14X,E21.14,7X,I2)                   00000390
                     STOP                                                                00000400
     30              END                                                                 00000410
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
46 TEST1

VARIABLES	SN	TYPE	RELOCATION				
1044	A	REAL	ARRAY	224	C	REAL	ARRAY
3374	CHSIVP	REAL	ARRAY	216	I	INTEGER	
215	IA	INTEGER		220	IARG	INTEGER	
5724	ITYPE	INTEGER	ARRAY	217	J	INTEGER	
214	N	INTEGER		222	N1	INTEGER	
223	N2	INTEGER		221	PRE	REAL	
3444	VECT	REAL	ARRAY	3324	VP	REAL	ARRAY

FILE NAMES	MODE						
0	INPUT	20	OUTPUT	FMT	0	TAPE5	20 TAPE6

EXTERNALS	TYPE	ARGS			
AMONIT		7	CHSIQR		10

```
      1                SUBROUTINE AMONIT(VP,CHSIVP,A,C,IA,N,PRE)                  00001440
                       DIMENSION VP(IA,1),CHSIVP(IA,1),A(IA,1),C(IA,1)            00001450
                       DO 1 I=1,N                                                 00001460
                       IF(CHSIVP(I,1).GT.PRE)GOTO 1                               00001470
      5                DVAL=VP(I,1)/(10**CHSIVP(I,1))*2.48                        00001480
                       CALL AOPTIM(VP(I,1),DVAL,A,C,IA,N,PRE)                     00001490
                       CHSIVP(I,1)=ALOG10(VP(I,1)/DVAL)                           00001500
                  1    CONTINUE                                                   00001510
                       RETURN                                                     00001520
     10                END                                                        00001530
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 AMONIT

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	C	REAL	ARRAY	F.P.
0	CHSIVP	REAL	ARRAY	F.P.	47	DVAL	REAL		
46	I	INTEGER			0	IA	INTEGER		F.P.
0	N	INTEGER		F.P.	0	PRE	REAL		F.P.
0	VP	REAL	ARRAY	F.P.					

EXTERNALS	TYPE	ARGS			
ALOG10	REAL	1	LIBRARY	AOPTIM	7

STATEMENT LABELS
32 1

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
7	1	I	3 8	26B	EXT REFS

STATISTICS
PROGRAM LENGTH 65B 53
55000B SCM USED

NO	PARTIE REELLE	CHIFF.SIGN.	PARTIE IMAGINAIRE	CHIFF.SIGN.
1	.45946781371747E+02	12	0.	15
2	.31913919887477E+02	12	0.	15
3	.22141483257827E+02	12	0.	15
4	.14899173038189E+02	12	0.	15
5	.95138427238770E+01	12	0.	15
6	.56326922741220E+01	12	0.	15
7	.30237870566649E+01	12	0.	15
8	.14687177522742E+01	9	0.	15
9	.68086603952477E+00	6	0.	15
10	.33071182132329E+00	4	0.	15
11	.17748112279988E+00	2	0.	15
12	.10573908610225E+00	1	0.	15
13	.67259334183086E-01	0	0.	15
14	.48953845881071E-01	0	0.	15
15	.25909307339195E-01	0	0.	15
16	.22682080952330E-01	0	0.	15

Tableau IX

IV.2.10 - Utilisation du polynôme caractéristique

Nous avons vu que, dans certains cas, la résolution du polynôme caractéristique pouvait aboutir à une meilleure précision sur les valeurs propres que la méthode QR.

Nous sommes conscients que le calcul du polynôme peut entraîner une imprécision sur les coefficients. Néanmoins, si nous prenons en compte le bruit numérique sur les coefficients, nous pouvons aussi estimer le bruit numérique des valeurs propres obtenues par cette méthode.

Si nous améliorons maintenant les résultats obtenus par la méthode QR à l'aide de l'utilisation du polynôme caractéristique tout en contrôlant le bruit numérique de cette seconde méthode, les résultats finaux seront les meilleurs des deux méthodes.

Puisqu'il s'agit d'améliorer et non de calculer, nous pouvons supposer être suffisamment proches de la solution pour pouvoir appliquer directement la méthode de Newton.

L'algorithme que nous allons définir sera alors rapide.

Dans ce cadre, nous pouvons dire que la fonction $P(\lambda)$ est monotone sur l'intervalle $I_0 = [\lambda_0 - \varepsilon, \lambda_0 + \varepsilon]$, λ_0 étant la valeur propre approchée, $P(\lambda)$ le polynôme caractéristique et ε tel que $P(\lambda)$ a un zéro sur I_0.

Il est possible alors d'utiliser directement la méthode de Raphson-Newton, c'est-à-dire d'élaborer la suite $\lambda_0, \lambda_1, \ldots, \lambda_n, \lambda_{n+1}, \ldots$ tendant vers la valeur propre exacte voisine.

Ainsi que le montre la figure 1, le point λ_{n+1} correspond à l'intersection avec l'axe des abscisses de la tangente à la courbe $y = P(\lambda)$ au point λ_{n+1}.

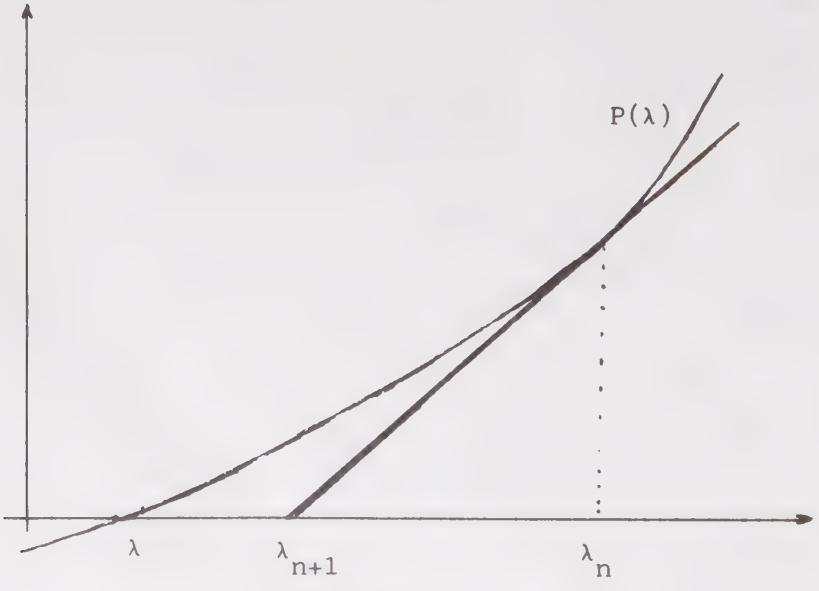


Figure 1

Cette tangente a pour équation :

$$y - P(\lambda_n) = P'(\lambda_n)\ (\lambda - \lambda_n) \qquad (4.2.60)$$

La formule de Newton s'obtient en faisant y = 0 ; elle s'écrit :

$$\lambda_{n+1} = \lambda_n - \frac{P(\lambda_n)}{P'(\lambda_n)} \qquad (4.2.61)$$

Notons que $\phi(\lambda) = \frac{P(\lambda)}{P'(\lambda)}$ s'annule avec $P(\lambda)$.

Expression analytique de $\phi(\lambda)$

$P(\lambda)$ s'écrit : $P(\lambda) = \sum_{i=0}^{n} a_i \lambda^i$. Nous en déduisons $P'(\lambda)$ par :

$$P'(\lambda) = \sum_{i=1}^{n} i\, a_i\, \lambda^{i-1} \qquad (4.2.62)$$

soit :

$$\phi(\lambda_n) = \frac{\sum_{i=0}^{n} a_i \lambda^i}{\sum_{i=1}^{n} i\, a_i\, \lambda^{i-1}} \qquad (4.2.63)$$

Nous arrêtons les itérations pour $\phi(\lambda_n) = 0$, c'est-à-dire lorsque le résultat signal/bruit du résultat numérique est inférieur à 1.

Nous prendrons encore par précaution : S/B < 2. Soit :

$$\mathrm{Log}_{10} \frac{|\bar{R}|}{\delta} < 0.3$$

Obtention des résultats R_1, R_2, R_3

Nous cherchons encore trois résultats représentatifs des résultats numériques possibles.

Nous avions vu que la méthode de multiplication est aisée de mise en oeuvre pour des fonctions linéaires :

$$f(k.x) = k\ f(x)$$

ou, pour des fonctions pseudo-linéaires :

$$f(k.x) = g(k)\ f(x)$$

Nous allons montrer que cette méthode est encore applicable pour des expressions polynomiales.

Soit $z_0 = k$ et la suite z_i définie par :

$$z_{i+1} = z_i \cdot x \qquad \text{pour } i = 0, \ldots, n-1$$

Alors

$$\sum_{i=0}^{n} a_i z_i = k \sum_{i=0}^{n} a_i x^i \tag{4.2.64}$$

La démonstration est triviale. De plus :

$$\frac{\sum_{i=0}^{n} a_i z_i}{\sum_{i=1}^{n} i\, a_i z_{i-1}} = \frac{\sum_{i=0}^{n} a_i x^i}{\sum_{i=1}^{n} i\, a_i x^{i-1}} \tag{4.2.65}$$

Nous avons donc trouvé un algorithme de calcul de $\phi(x)$ où la méthode de multiplication peut encore être appliquée. Le choix des valeurs k reste encore le même, c'est-à-dire 1., 3. et 5.

Qu'avons-nous fait ?

Nous avons substitué la fonction f(x) par la fonction h(y,x) avec y = 1, soit :

$$h(1,x) = f(x)$$

où h a la propriété suivante : il existe une fonction g telle que :

$$h(k,x) = g(k)\ h(1,x) = g(k)\ f(x)$$

Ceci ouvre un nouveau champ d'applications pour la méthode de multiplication. Encore faut-il prendre la précaution, comme nous l'avons fait en (4.2.64) de s'assurer que l'algorithme de h substituant celui de f modifie bien toutes les affectations machine après opération.

Enfin, par l'algorithme POLCAR, nous avons obtenu trois résultats P_1, P_2, P_3 représentatifs des résultats machine, P_1, P_2, P_3 symbolisant les coefficients du polynôme caractéristique.

Pour tenir compte des erreurs numériques sur les coefficients du polynôme, nous appliquerons le coefficient multiplicatif k_1 aux données P_1, k_2 pour P_2 et k_3 pour P_3.

Les trois résultats R_1, R_2, R_3 ainsi obtenus intégreront aussi bien le bruit numérique sur $\phi(x)$ que celui sur les coefficients du polynôme.

Afin d'éviter le risque de converger deux fois vers la même valeur propre, après chaque amélioration d'une valeur propre nous diviserons le polynôme par la racine améliorée. Ce risque de converger deux fois vers la même valeur propre peut intervenir dans le cas où certaines valeurs propres sont calculées avec une précision insuffisante.

Nous proposons le programme AMONIT permettant d'améliorer les valeurs propres d'une matrice A réelle. Dans cette version du programme, il est indispensable que toutes les valeurs propres soient réelles. Dans le cas où certaines valeurs propres seraient complexes, il serait nécessaire de se ramener à la méthode de Bairstow.

Toutefois, la présence de deux racines complexes conjuguées simplifiera l'évaluation des coefficients s et p du polynôme $\lambda^2 - s\lambda + p$ recherché. De plus, la méthode de Newton à une seule variable pourrait encore être utilisée sur les valeurs propres réelles.

IV.2.11 - Notice d'utilisation du programme AMONIT

Objet

Amélioration des valeurs propres par le calcul du polynôme caractéristique. Toutes les valeurs propres sont réelles.

Appel

```
CALL AMONIT(VP,CHSIVP,A,B,C,P,CP,IA,N)
```

Arguments d'entrée

VP Tableau dimensionné au moins à N contenant les valeurs propres réelles approchées de A.

A Tableau bidimensionné dont la première dimension est IA et la seconde au moins égale à N.

N Ordre de la matrice A.

IA Première dimension de A,B,C,P,CP.

Arguments de sortie

CHSIVP Tableau dimensionné au moins à N contenant le nombre de chiffres significatifs (Tableau réel) des valeurs propres améliorées. Valeurs non modifiées si aucune amélioration.

VP Valeurs propres améliorées.

B Tableau de mêmes dimensions que A. Contient la matrice inverse si elle existe.

C Tableau de travail de mêmes dimensions que A.

P Tableau bidimensionné de première dimension égale à IA et la seconde égale à 3. Contient les coefficients du polynôme caractéristique avec trois résultats issus de calculs différents.

$$P(\lambda) = \lambda^n - p_1 \lambda^{n-1} - \ldots - p_{n-1} \lambda - p_n$$

CP Tableau de dimensions au moins égales à N, contient le nombre de chiffres significatifs pour chacun des coefficients du polynôme. Tableau réel.

IV.2.12 - Programme FORTRAN AMONIT

```
      SUBROUTINE AMONIT(VP,CHSIVP,A,B,C,P,CP,IA,N)                      00000800
      DIMENSION VP(IA,1),CHSIVP(IA,1),A(IA,1),C(IA,1),P(IA,1),B(IA,1)   00000810
      CALL POLCAR(A,B,C,P,CP,IA,N)                                      00000820
      M=N                                                               00000830
      DO 1 L=1,N                                                        00000840
      I=N-L+1                                                           00000850
      K=0                                                               00000860
      X0=VP(I,1)                                                        00000870
    3 DX=PAS(X0,P,IA,M,CR)                                              00000880
      X=X0-DX                                                           00000890
      K=K+1                                                             00000900
      IF(X.EQ.X0)GOTO 4                                                 00000910
      X0=X                                                              00000920
      CDXEND=CR                                                         00000930
      IF(K.LE.30)GOTO 3                                                 00000940
    4 VP(I,1)=X                                                         00000950
      CHSIVP(I,1)=CDXEND                                                00000960
      M=M-1                                                             00000970
      DO 11 IS=1,3                                                      00000980
      P(1,IS)=P(1,IS)-X                                                 00000990
      DO 11 J=2,M                                                       00001000
   11 P(J,IS)=P(J,IS)+X*P(J-1,IS)                                       00001010
    1 CONTINUE                                                          00001020
      RETURN                                                            00001030
      END                                                               00001040
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 AMONIT

VARIABLES	SN	TYPE		RELOCATION					
0	A	REAL	ARRAY	F.P.	0	B	REAL	ARRAY	F.P.
0	C	REAL	ARRAY	F.P.	132	CDXEND	REAL		
0	CHSIVP	REAL	ARRAY	F.P.	0	CP	REAL		F.P.
130	CR	REAL			127	DX	REAL		
124	I	INTEGER			0	IA	INTEGER		F.P.
133	IS	INTEGER			134	J	INTEGER		
125	K	INTEGER			123	L	INTEGER		
122	M	INTEGER			0	N	INTEGER		F.P.
0	P	REAL	ARRAY	F.P.	0	VP	REAL	ARRAY	F.P.
131	X	REAL			126	X0	REAL		

EXTERNALS	TYPE	ARGS		
PAS	REAL	5	POLCAR	7

STATEMENT LABELS

0	1	37	3	51	4
0	11				

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
31	1	L	5 23	50B	EXT REFS NOT INNER
63	11	IS	19 22	13B	NOT INNER

```
      1              FUNCTION PAS(X,P,IA,N,CR)                                 00000440
                     DIMENSION P(IA,3),CMUL(3),R(3),RX(3)                      00000450
                     COMMON/WXTST/CMIN,CMAX                                    00000460
                     LOGICAL RETOUR                                            00000470
      5              DATA CMUL/1.,3.,5./                                       00000480
                     CMIN=0.3                                                  00000490
                     CMAX=1.                                                   00000500
                     DO 1 IS=1,3                                               00000510
                     PUISS=X*CMUL(IS)                                          00000520
     10              F=-P(N,IS)*CMUL(IS)                                       00000530
                     NM1=N-1                                                   00000540
                     FD=-P(NM1,IS)*CMUL(IS)                                    00000550
                     NM2=N-2                                                   00000560
                     DO 2 I=1,NM2                                              00000570
     15              L=N-I                                                     00000580
                     LM1=L-1                                                   00000590
                     F=F-P(L,IS)*PUISS                                         00000600
                     FD=FD-(I+1)*P(LM1,IS)*PUISS                               00000610
                     PUISS=PUISS*X                                             00000620
     20            2 CONTINUE                                                  00000630
                     F=F-P(1,IS)*PUISS                                         00000640
                     FD=FD+N*PUISS                                             00000650
                     PUISS=PUISS*X                                             00000660
                     F=F+PUISS                                                 00000670
     25          C   PRINT *,X,PUISS,F,FD                                      00000680
                     R(IS)=0.                                                  00000690
                     IF(FD.NE.0.)R(IS)=F/FD                                    00000700
                     RX(IS)=X-R(IS)                                            00000710
                 I   CONTINUE                                                  00000720
     30              CALL WXTEST(RX,3,CR,DUM,RETOUR)                           00000730
                     CALL WXTEST(R,3,DUM,RM,RETOUR)                            00000740
                     IF(RETOUR)RM=0.                                           00000750
                     PAS=RM                                                    00000760
                 C   PRINT *,R(1),R(2),R(3)                                    00000770
     35              RETURN                                                    00000780
                     END                                                       00000790
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 PAS

VARIABLES	SN	TYPE	RELOCATION						
1	CMAX	REAL	WXTST	0	CMIN	REAL		WXTST	
123	CMUL	REAL	ARRAY	0	CR	REAL		F.P.	
121	DUM	REAL		112	F	REAL			
114	FD	REAL		116	I	INTEGER			
0	IA	INTEGER	F.P.	110	IS	INTEGER			
117	L	INTEGER		120	LM1	INTEGER			
0	N	INTEGER	F.P.	113	NM1	INTEGER			
115	NM2	INTEGER		0	P	REAL	ARRAY	F.P.	
106	PAS	REAL		111	PUISS	REAL			
126	R	REAL	ARRAY	107	RETOUR	LOGICAL			

```
      1            PROGRAM TEST1(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)              00000120
                   DIMENSION C(20,20),A(20,20,3),VP(20,2),CHSIVP(20,2),              00000130
                  1 VECT(20,20,3),ITYPE(20),B(20,20),AA(20,20),P(20,3),CP(20,3)      00000140
                   N=16                                                              00000150
      5            IA=20                                                             00000160
                   DO 1 I=1,N                                                        00000170
                   DO 2 J=1,I                                                        00000180
                2  A(I,J,1)=0.                                                      00000190
                   DO 3 J=I,N                                                        00000200
     10            A(I,J,1)=N-J+1                                                    00000210
                   IF(I.EQ.1)GOTO 1                                                  00000220
                   A(I,I-1,1)=N-I+1                                                  00000230
                1  CONTINUE                                                         00000240
                   DO 8 I=1,N                                                        00000250
     15            DO 5 J=1,N                                                        00000260
                   VECT(I,J,1)=0.                                                    00000270
                5  AA(I,J)=A(I,J,1)                                                  00000280
                8  VECT(I,I,1)=0.                                                   00000290
                   IARG=0                                                            00000300
     20            CALL CHSIGR(A,N,IA,1,N,VP,CHSIVP,VECT,ITYPE,IARG)                 00000310
                   PRE=10.                                                           00000320
                   CALL AMONIT(VP,CHSIVP,AA,B,C,P,CP,IA,N)                           00000330
                   PRINT 7                                                           00000340
                7  FORMAT(1H1,5X,2HNO,5X,13HPARTIE REELLE,7X,11HCHIFF.SIGN.,10X,     00000350
     25            1   17HPARTIE IMAGINAIRE,3X,11HCHIFF.SIGN.)                       00000360
                   DO 4 I=1,N                                                        00000370
                   N1=CHSIVP(I,1)                                                    00000380
                   N2=CHSIVP(I,2)                                                    00000390
                4  PRINT 6,I,VP(I,1),N1,VP(I,2),N2                                  00000400
     30         6  FORMAT(/2X,I5,3X,E21.14,7X,I2,14X,E21.14,7X,I2)                   00000410
                   STOP                                                              00000420
                   END                                                               00000430
```


SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
46 TEST1

VARIABLES	SN	TYPE	RELOCATION					
1052	A	REAL	ARRAY	6576	AA	REAL	ARRAY	
5756	B	REAL	ARRAY	232	C	REAL	ARRAY	
3402	CHSIVP	REAL	ARRAY	7512	CP	REAL	ARRAY	
224	I	INTEGER		223	IA	INTEGER		
226	IARG	INTEGER		5732	ITYPE	INTEGER	ARRAY	
225	J	INTEGER		222	N	INTEGER		
230	N1	INTEGER		231	N2	INTEGER		
7416	P	REAL	ARRAY	227	PRE	REAL		
3452	VECT	REAL	ARRAY	3332	VP	REAL	ARRAY	

FILE NAMES							
0	INPUT	20	OUTPUT	FMT	0	TAPE5	20 TAPE6

IV.2.13 - Résultats numériques

Nous effectuons une amélioration des valeurs propres calculées par l'algorithme QR comme en IV.2.9.

Nous observons que les résultats sont comparables à ceux déjà obtenus en III.3.2 par la méthode de Bairstow.

Nous remarquons que le nombre d'opérations élémentaires est surtout dépendant de l'algorithme de calcul du polynôme caractéristique. L'amélioration des valeurs propres ne nécessite en effet qu'un nombre négligeable d'opérations à chaque itération.

NO	PARTIE REELLE	CHIFF.SIGN.	PARTIE IMAGINAIRE	CHIFF.SIGN.
1	.45946781371747E+02	14	0.	15
2	.78613853985850E+02	14	0.	15
3	.22141483257756E+02	13	0.	15
4	.14899173038257E+02	13	0.	15
5	.95138427238603E+01	13	0.	15
6	.56326922741353E+01	13	0.	15
7	.30237870566691E+01	13	0.	15
8	.14687177523957E+01	14	0.	15
9	.68086601279862E+00	13	0.	15
10	.33071111862651E+00	13	0.	15
11	.17753499593670E+00	13	0.	15
12	.10510999908478E+00	13	0.	15
13	.67117819051814E-01	13	0.	15
14	.45164092592838E-01	12	0.	15
15	.31334289348494E-01	12	0.	15
16	.21764310146311E-01	13	0.	15

IV.3 - AMELIORATION DES VECTEURS PROPRES

IV.3.1 - Notion de résidu minimal

Qu'est-ce qu'un vecteur propre ? C'est le vecteur solution du système :

$$(A - \lambda I)\, V = 0 \qquad (4.3.1)$$

Si λ n'est qu'une valeur proche approchée, une telle solution n'existe pas en dehors de $V = 0$. Si λ est une valeur propre exacte, il existe une infinité de solutions. Ces solutions forment un hyperespace de dimensions supérieures ou égales à 1 selon la dimension de l'espace propre associé à la valeur λ, noté E_λ.

Soit λ_0 une valeur propre calculée approchant la valeur propre théorique λ. Comment calculer un vecteur V_0 le plus proche possible du vecteur propre théorique V ?

Si V_0 a été calculé avec une valeur λ_0, que λ_0 ait été améliorée, comment améliorer V_0 ?

Vu l'infinité des solutions possibles, nous pourrons déjà nous limiter à des vecteurs V unitaires pour la norme Euclidienne, c'est-à-dire $||V||_2 = 1$, soit $\sum_{i=1}^{n} |v_i|^2 = 1$, où les v_i sont les projections sur les vecteurs de la base canonique de $\mathbb{K}^n$ (on se limitera ici à $\mathbb{R}^n$ ou $\mathbb{C}^n$) pour $V \in \mathbb{K}^n$. Si $\dim E_\lambda = 1$, il n'existe plus que deux solutions V_1 et V_2 au système (4.3.1), qui sont liées par $V_1 = - V_2$. De plus, nous éliminons la solution triviale $V = 0$.

Soit λ_0 et V_0 valeur et vecteur approchés des solutions propres de (4.3.1) et notons :

$$R_0 = (A - \lambda_0 I)\, V_0 \qquad (4.3.2)$$

le résidu du système (4.3.1).

Il est clair que pour $\lambda_0 \neq \lambda$ ou $V_0 \neq V$ nous avons $R_0 \neq 0$, $R_0 = 0$ dans le cas contraire.

La solution V_0 peut alors se ramener à trouver R_0 le plus petit possible, c'est-à-dire trouver V_0 tel que $||R_0||_2$ soit minimum, et ceci pour $||V_0||_2 = 1$.

Considérons alors la fonctionnelle :

$$J(V_0) = ||R_0||_2^2 \qquad (4.3.2)$$

Il reste à montrer que :

1) des minimums existent,

2) ces minimums sont bien ce que nous cherchons, quel que soit le minimum trouvé.

Pour ceci, nous aurons besoin des notions de normes vectorielles et de normes matricielles dans les espaces pré-hilbertiens. Nous allons donc en rappeler les propriétés.

Soit X, un espace vectoriel ($\mathbb{R}^n$ ou $\mathbb{C}^n$) sur le corps $\mathbb{K}$ des scalaires ($\mathbb{R}$ ou $\mathbb{C}$). Une norme vectorielle est une application $||.|| : X \to \mathbb{R}$ qui, à tout vecteur $x \in X$, associe un nombre réel noté $||x||$ et qui vérifie les propriétés suivantes :

$$\|x\| > 0 \Longleftrightarrow x \neq 0 \quad \text{et} \quad \|0\| = 0$$

$$\|\alpha\, x\| = |\alpha|\ \|x\| \qquad \forall\, \alpha \in \mathbb{K}$$

$$\|x + y\| \leq \|x\| + \|y\| \quad \text{(inégalité triangulaire)}$$

Soit $\mathcal{M}$ l'anneau des matrices carrées nxn sur le corps $\mathbb{K}$ des scalaires. Une norme matricielle est une application $||.|| : \mathcal{M} \to \mathbb{R}$ qui à toute matrice $A \in \mathcal{M}$ associe un nombre réel noté $||A||$ et qui vérifie les propriétés suivantes :

$$\|A\| > 0 \iff A \neq 0 \quad \text{et} \quad \|0\| = 0$$

$$\|\alpha\, A\| = |\alpha|\ \|A\|, \qquad \alpha \in \mathbb{K}$$

$$\|A+B\| \leq \|A\| + \|B\|$$

$$\|A.B\| \leq \|A\|.\|B\|$$

A partir d'une norme vectorielle définie sur un espace vectoriel X, on peut construire une norme matricielle induite (par la norme vectorielle en question) de la façon suivante ; nous posons :

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\| = \sup_{\|x\| \leq 1} \|Ax\|$$

$\mathbb{K}^n$($\mathbb{R}^n$ ou $\mathbb{C}^n$) est un espace pré-hilbertien.

Nous pouvons définir un produit scalaire $\langle u, v^* \rangle$ noté encore $v^*.u$ tel que :

$$\|x\|_2^2 = \langle x, x^* \rangle = x^*.x \qquad (4.3.3)$$

x^* étant le vecteur auto-adjoint de x qui peut être défini dans $\mathbb{C}^n$ par :

$$x^* = (x_i)^* = (\bar{x}_i)^t \qquad \bar{x}_i \text{ étant le conjugué de } x_i$$

$(x_i)^t$ vecteur transposé (vecteur ligne) du vecteur (x_i). Pour $x \in \mathbb{R}^n$, nous aurons simplement :

$$x^* = (x_i)^t$$

Nous définissons de même A^*, matrice auto-adjointe de A, par :

$$A^* = (a_{ij})^* = (\bar{a}_{ij})^t = (\bar{a}_{ji})$$

Nous pouvons alors écrire la fonctionnelle $J(V_0)$ définie en (4.3.2) sous la forme :

$$J(V_0) = V_0^*(A - \lambda_0 I)^* (A - \lambda_0 I) V_0 \qquad (4.3.4)$$

Si $B = (A - \lambda_0 I)^*(A - \lambda_0 I)$, il est facile de montrer que B est une matrice symétrique réelle à coefficients positifs ou nuls. Nous écrirons encore :

$$J(V_0) = V_0^* B V_0 \qquad (4.3.5)$$

Soit maintenant $V_0 \in \mathbb{C}^n$. Posons :

$$V_0 = V_{OR} + i V_{OI}$$

où V_{OR} et $V_{OI} \in \mathbb{R}^n$ et si nous développons $J(V_0)$ nous obtenons :

$$J(V_0) = (V_{OR}^t - i V_{OI}^t) B (V_{OR} + i V_{OI})$$

$$J(V_0) = V_{OR}^t B V_{OR} - i V_{OI}^t B V_{OR} + i V_{OR}^t B V_{OI} + V_{OI}^t B V_{OI}$$

Or $V_{OI}^t B V_{OR} = V_{OR}^t B V_{OI}$. D'où :

$$J(V_0) = V_{OR}^t B V_{OR} + V_{OI}^t B V_{OI} \qquad (4.3.6)$$

Si U est un vecteur de $\mathbb{R}^{2n}$ défini par $U = (u_i)$ tel que :

$$\left.\begin{array}{l} u_i = v_{ORi} \\ u_{i+n} = v_{OIi} \end{array}\right| \quad \text{pour } i = 1,n$$

alors l'étude de $J(V_0)$ se ramène à celle de $F(U)$, où :

$$F(U) = U^t C\, U = J(V_0) \qquad (4.3.7)$$

avec $C = \begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}$, ramenant ainsi le cas complexe au cas réel.

Nous désignerons indifféremment $F(U)$ ou $J(V_0)$ pour $V_0 \in \mathbb{R}^n$ ou $\mathbb{C}^n$.

Existence d'un minimum

Soit $D = \{ V_0 \in \mathbb{K}^n \mid ||V_0||_2 = 1 \}$ (cercle unité).

D est un compact de $\mathbb{K}^n$.

Par ailleurs, $F(U)$ est d'après (4.3.7) une fonctionnelle quadratique, c'est donc bien une fonction continue de $\mathbb{R}^{2n}$ ou $\mathbb{R}^n$ dans $\mathbb{R}$.

En application du théorème (|7|, p.38), l'image de $F(U)$ donc de $J(V_0)$ est compacte.

$J(V_0)$ accepte donc au moins un minimum pour $V_0 \in D$.

Nous allons maintenant définir un algorithme de minimisation sous contrainte et montrer a posteriori que cet algorithme converge bien vers un vecteur propre.

Nous avons pour cela encore besoin de définir H(V) (Hyperespace orthogonal à un vecteur V) par $H(V) = \{ W \in \mathbb{K}^n \mid W^*.V = 0 \text{ et } ||W||_2 \neq 0 \}$.

IV.3.2 - Une méthode de minimisation sous contrainte

Nous proposons le lemme suivant :

Lemme 1

$\forall \lambda \in \mathbb{K}$, $V \in D$, $W \in H(V)$, $\eta \in \mathbb{R}$, il existe $\theta \in \mathbb{R}$ unique tel que :

$$J\left(\frac{V + \eta W}{\|V + \eta W\|}\right) \geqslant J\left(\frac{V + \theta W}{\|V + \theta W\|}\right) \qquad (4.3.8)$$

ou bien W vérifie :

$$J\left(\frac{W}{\|W\|}\right) \leqslant J(V) \qquad (4.3.9)$$

Démonstration

Soit $\phi(\theta) = J\left(\frac{V + \theta W}{\|V + \theta W\|}\right) - J(V)$

$$\phi(\theta) = \frac{1}{\|V + \theta W\|^2} \left[(V+\theta W)^*(A-\lambda I)^*(A-\lambda I)(V+\theta W)\right] - J(V).$$

Or :

$$\|V+\theta W\|^2 = (V+\theta W)^*(V+\theta W) = V^*.V+\theta W^*.V+\theta V^*.W+\theta^2\, W^*.W$$

$$= 1 + \theta^2 \|W\|^2$$

d'où :

$$\phi(\theta) = \frac{1}{1+\theta^2\|W\|^2}\left[\theta^2 J(W)+\theta(V^*(A-\lambda I)^*(A-\lambda I)W + W^*(A-\lambda I)^*(A-\lambda I)V)\right.$$

$$\left. + J(V) - (1+\theta^2\|W\|^2)J(V)\right]$$

Nous posons :

$$b = V^*(A-\lambda I)^*(A-\lambda I)W$$

$$W^*(A-\lambda I)^*(A-\lambda I)V = \bar{b}$$

et $b + \bar{b} = 2$ Réel(b) (Partie réelle).

d'où :

$$\phi(\theta) = \frac{1}{1+\theta^2 \|W\|^2} [\theta^2 (J(W) - \|W\|^2 J(V)) + 2\theta \text{ Réel } (b)]$$

Posons encore :

$e = J(W) - g\, J(V)$ avec $g = \|W\|^2$

et $f =$ Réel (b)

et nous obtenons :

$$\boxed{\phi(\theta) = \frac{\theta^2 e + 2\theta f}{1 + \theta^2 g}} \qquad (4.3.10)$$

$$\phi'(\theta) = \frac{(2\theta e + 2f)(1+\theta^2 g) - 2\theta g(\theta^2 e + 2\theta f)}{(1+\theta^2 g)^2}$$

d'où $\phi'(\theta) = 0 \iff$ $\boxed{-\theta^2 fg + \theta e + f = 0}$ (4.3.11)

Nous vérifions immédiatement que le déterminant de cette équation du second degré en θ ;

$$\Delta = e^2 + 4f^2 g \quad \text{est toujours positif}$$

Pour $f \neq 0$ (4.3.11) admet les racines :

$$\boxed{\theta = \frac{e \pm \sqrt{e^2 + 4f^2 g}}{2 f g}} \qquad (4.3.12)$$

L'étude de $\phi(\theta)$ montre alors que $\forall f \neq 0$:

$$\theta = \frac{e - \sqrt{e^2 + 4f^2 g}}{2 f g} \qquad (4.3.13)$$

est le minimum unique de $\phi(\theta)$.

Pour f = 0, (4.3.11) se ramène à :

$$\theta e = 0$$

$\Longleftrightarrow \theta = 0$ ou $e = 0$

et si $e = 0 \Longleftrightarrow J(W) - gJ(V) = 0$.

$$\text{Or : } J\left(\frac{W}{\|W\|}\right) = \frac{1}{\|W\|^2} J(W) = \frac{1}{g} J(W) \text{ ; d'où :}$$

$$e = 0 \Longleftrightarrow J(V) = J\left(\frac{W}{\|W\|}\right) \qquad (4.3.15)$$

et nous constaterons que $\phi(\theta) = 0$. $\forall\ \theta \in \mathbb{R}$.

Un tel lemme n'est intéressant que si la minimisation de J(V) peut se ramener à un nombre fini de directions W_i. Il en découle le théorème suivant.

Théorème 1

Soit $\{W_i\}_{i \in I}$ une famille de vecteurs de H(V) recouvrant H(V), c'est-à-dire que $\forall\ W \in H(V)$, Il existe une famille de scalaires $\{a_i\}_{i \in I}$ avec $a_i \in \mathbb{R}$, $\forall_i \in I$ tel que :

$$W = \sum_{i \in I} a_i W_i \qquad (4.3.16a)$$

Alors une condition nécessaire et suffisante pour que V soit un minimum de J sur D est que $\forall i \in I$, $\forall\ \theta \in \mathbb{R}$.

$$J\left(\frac{V + \theta W_i}{\|V + \theta W_i\|}\right) \geqslant J(V) \qquad (4.3.16b)$$

Démonstration

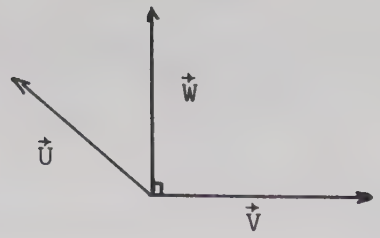
Que (4.3.16b) soit une condition nécessaire est immédiat.

V minimise J sur D si $\forall\ U \in D$:

$$J(U) \geqslant J(V)$$

Le cas où $U = \pm V$ n'offrant pas d'intérêt, considérons l'intersection du plan de dimension 2 défini par $\vec{U}$ et $\vec{V}$, et de l'hyperespace H(V).

Cette intersection est de dimension 1 et peut être définie par un vecteur unitaire $\vec{W}$:



Nous définissons alors $\vec{U}$ par :

$$U = \frac{aV + bW}{||aV + bW||}$$

1) $U \notin H(V)$: alors $a \neq 0$, et :

$$U = \pm \frac{V + b/a\ W}{||V + b/a\ W||} = \pm \frac{V + \theta W}{||V + \theta W||}$$

En posant $\theta = b/a$, et comme $J(U) = J(-U)$, nous obtenons bien :

$$J(U) = J\left(\frac{V + \theta W}{||V + \theta W||}\right)$$

1) $U \in H(V)$, alors $U = \pm W$

Nous allons alors montrer que si $\forall\ U \notin H(V)$, $J(U) > J(V)$, alors $J(U) \geqslant J(V)$ pour $U \in H(V)$.

Soit la suite de vecteurs U_n définie par :

$$U_n = \frac{V + nW}{||V + nW||}$$

Comme V et W sont unitaires et orthogonaux :

$$U_n = \frac{V + nW}{\sqrt{1 + n^2}}$$

$$\text{d'où} \quad \|U_n - W\|^2 = \frac{1}{1+n^2} \|V + n W - \sqrt{1+n^2}\ W\|^2$$

$$= \frac{1}{1+n^2}(1 + (n - \sqrt{1+n^2})^2)$$

$$\text{pour } n \to +\infty, \sqrt{1+n} = \sqrt{n^2(1+\frac{1}{n^2})} = n\sqrt{1+\frac{1}{n^2}} \simeq n(1 + \frac{1}{2}\frac{1}{n^2})$$

$$\text{d'où} \quad \|U_n - W\|^2 \simeq \frac{1 - \frac{1}{2}\frac{1}{n^2}}{1+n^2} \xrightarrow[n \to \infty]{} 0.$$

U_n converge donc vers W, par continuité $J(U_n)$ converge aussi vers $J(W)$. Or, l'image réciproque de l'ensemble G défini par $G = \{U \in D \mid J(U) \geqslant J(V)\}$ est un fermé de D. C'est donc aussi un compact, et l'image de G par J est encore un compact que nous appellerons F.

Il est évident que si $\forall n \in \mathbb{N}$, $\forall\ U_n \in G$, donc si $\forall J(U_n) \in F$, nous avons

$$\lim_{n \to +\infty} U_n = W \in G$$

et

$$\lim_{n \to +\infty} J(U_n) \in F$$

alors $\forall n \in \mathbb{N}$ $J(U_n) \geqslant J(V) \Longrightarrow J(W) \geqslant J(V)$. Ceci achève de montrer que V est un minimum de J sur D équivaut à $\forall\ W \in H(V)$, $\forall\ \theta \in \mathbb{R}$:

$$J\left(\frac{V + \theta W}{\|V + \theta W\|}\right) \geqslant J(V) \qquad (4.3.17)$$

Montrons que ceci est encore équivalent à $\forall i \in I$, $\forall\ \theta \in \mathbb{R}$.

$$J\left(\frac{V + \theta W_i}{\|V + \theta W_i\|}\right) \geqslant J(V) \qquad (4.3.18)$$

Si nous nous ramenons à la démonstration du lemme précédent, (4.3.17) est équivalent à ce que $\phi(0)$ soit un minimum, ou encore à $\phi'(0) = 0$, ou encore à $f = 0$ (voir 4.3.11).

$$\text{Or } f = 0 \ \forall\, W \in H(V) \Longleftrightarrow \text{Réel}(V^*(A-\lambda I)^*(A-\lambda I)W) = 0$$

$$\forall\, W \in H(V)$$

$$\Longleftrightarrow \sum_{i\in I} \text{Reel}\,(a_i V^*(A-\lambda I)^*(A-\lambda I)W_i) = 0$$

$$\forall\, W \in H(V) \quad \text{avec } W = \sum_{i\in I} a_i\, W_i$$

$$\Longleftrightarrow \text{Reel}\,(V^*(A-\lambda I)^*(A-\lambda I)W_i) = 0 \quad \forall i \in I$$

et les mêmes raisons que pour (4.3.17) montrent que c'est encore équivalent à (4.3.18).

Essayons maintenant de définir cette famille $\{W_i\}_{i\in I}$ recouvrant H(V) avec $W_i \in H(V)$, $\forall i \in I$.

Lemme 2

Soit $\{E_i\}_{i\in I}$, la base canonique de $\mathbb{K}^n$; alors, si nous définissons W_i tel que :

$$W_i = E_i - (V^*.E_i)V \quad \forall i \in I \tag{4.3.19}$$

la famille $\{W_i\}_{i\in I}$ vérifie bien les conditions du théorème 1.

Démonstration

1) $W_i \in H(V)$ $\forall i \in I$;

En effet : $V^*.W_i = V^*.(E_i - (V^*.E_i)V)$

$= V^*.E_i - (V^*.E_i)\ V^*.V$

$= 0$ puisque $V^*.V = 1$.

2) Si $U \in H(V)$, il existe $i \in I$ tel que $U^*.W_i \neq 0$, ou bien $U = 0$.

En effet : $U^*.W_i = U^*(E_i - (V^*.E_i)V)$

$= U^*.E_i - (V^*.E_i)\ U^*.V$

$= U^*.E_i$

Or $U^*.E_i = 0$ $\forall i \in I \Longleftrightarrow U = 0$.

La base canonique pour $\mathbb{R}^n$ pourra être définie par $E_i = (e_{ij})$ avec $e_{ij} = 0$ pour $i \neq j$ et $e_{ii} = 1$, et avec $i = 1,n$ soit $I = \{1,\ldots,n\}$ et $j = 1,n$.

La base canonique sur $\mathbb{C}^n$ sera définie par $e_{ij} = 0$ pour $(i \neq 2j-1,\ 2\ )$, $e_{2j-1,j} = 1$, $e_{2j,j} = (\sqrt{-1})$. Nous aurons alors $j \in [1,n]$ et $i \in [1,2n]$ soit $I = \{1,\ldots,2n\}$.

Algorithme de descente proposé

A l'itération p nous disposons du vecteur V^p et nous posons $V_0^p = V^p$.

Nous définissons $W_1 \in H(V_0^p)$ par :

$$W_1 = E_1 - (V_0^{p*}.E_1)V_0^p \qquad (4.3.20)$$

et θ_1 solution de (4.3.13), c'est-à-dire comme la valeur $\theta \in \mathbb{R}$ telle que :

$$J\left(\frac{V_0^p + \theta W_1}{\|V_0^p + \theta W_1\|}\right) \text{ soit minimum}$$

Nous posons alors :

$$V_1^p = \frac{V_0^p + \theta_1 W_1}{\|V_0^p + \theta_1 W_1\|} \qquad (4.3.21)$$

et $W_2 \in H(V_1^p)$ par $W_2 = E_2 - (V_1^{p*}.E_2)V_1^p$ (4.3.22)

Nous trouvons alors une solution θ_2.

A l'étape i, nous obtiendrons :

$$W_i = E_i - ((V_{i-1}^p)^*.\ E_i)\ V_i^p \qquad (4.3.23)$$

$$V_i^p = \frac{V_{i-1}^p + \theta_i W_i}{\|V_{i-1}^p + \theta_i W_i\|} \qquad (4.3.24)$$

et ceci jusqu'à l'étape m, avec $m = n$ si $\mathbb{K}^n = \mathbb{R}^n$, et $m = 2n$ si $\mathbb{K}^n = \mathbb{C}^n$.

Afin de limiter un ralentissement de la convergence liée à une descente suivant une arête, nous proposons l'étape supplémentaire suivante.

Soit $\vec{U} = V_m^p - V_o^p$, $\vec{U}$ approxime la direction de la plus grande pente. Soit S, la projection de $\vec{U}$ sur $H(V_m^p)$, S sera alors donné par :

$$S = (V_m^{p*}.V_o^p)\ V_m^p - V_o^p \qquad (4.3.25)$$

Nous chercherons encore θ_{m+1} minimisant :

$$J\left(\frac{V_m^p + \theta S}{\|V_m^p + \theta S\|}\right) \qquad (4.3.26)$$

puis nous poserons :

$$V^{p+1} = \frac{V_m^p + \theta_{m+1} S}{\|V_m^p + \theta_{m+1} S\|} \qquad (4.3.27)$$

et nous passerons ainsi à l'itération suivante.

Le vecteur V_o^1 initial sera un vecteur approché du vecteur solution V de :

$$(A - \lambda I)V = 0$$

Nous verrons qu'il peut être choisi quelconque.

Etudions maintenant les problèmes de convergence.

IV.3.3 - Convergence de la méthode

Si nous considérons la suite V^p que nous venons de définir au paragraphe précédent, la suite $J(V^p)$ forme une suite décroissante bornée inférieurement par une valeur $a \geqslant 0$.

Si, pour une valeur p, $J(V^{p+1}) = J(V^p)$, c'est que pour l'itération p, toutes les valeurs θ_i, i = 1,m trouvées, sont nulles et donc $V^{p+1} = V^p$.
Et donc $J(V^{p+q}) = J(V^p)$, $\forall q \in \mathbb{N}$.

Par application du théorème 1, V^p est un minimum de J sur D.

Par ailleurs, si V_1, V_2 sont deux minimums sur D, alors $J(V_1) = J(V_2)$ sinon nous avons vu que dans le cas général, nous trouverons W tel que :

$V_2 = \frac{V_1 + \theta W}{\|V_1 + \theta W\|}$ et l'existence d'un θ unique minimisant $J(\frac{V_1 + \theta W}{\|V_1 + \theta W\|})$. Ceci implique alors que $\theta = 0$. Dans les autres cas, nous pouvons nous ramener à $J(V_1) = J(V_2)$ (démonstration du théorème 1).

D'autre part, cette recherche des minimums peut se ramener à la recherche sur une autre famille $\{W_i\}_{i \in I}$ recouvrant H(V).

Il s'ensuit que tous les minimums V de J sont tels que les valeurs J(V) sont identiques.

Si λ est une valeur propre exacte

V est minimum $\Longleftrightarrow$ $J(V) = 0 \Longleftrightarrow (A-\lambda I)V = 0$

$\Longleftrightarrow$ V Vecteur propre.

Donc la suite $J(V^p)$ est strictement décroissante ou nulle. Nous avons bien convergence. Et ceci quelle que soit la dimension de l'espace propre associé à λ, noté E_λ.

Remarque : s'il existe $W \in H(V)$ tel que $J(\frac{W}{\|W\|}) = J(V)$, c'est donc que W est aussi un vecteur propre et que dim $E_\lambda > 1$.

Si λ est une valeur propre approchée

Nous avons encore une convergence vers un minimum. Avons-nous une infinité de solutions ? Ceci peut se produire s'il existe un vecteur $W \in H(V)$, V étant le minimum trouvé, tel que $J(\frac{W}{\|W\|}) = J(V)$.

Il faudrait alors encore montrer que dim $E_\lambda > 1$ et que tout vecteur $W + \theta W$ est un vecteur propre approché $\forall \theta \in \mathbb{R}$.

Nous allons montrer qu'il n'existe pas de vecteur W dans le cas général.

Soit $T = \frac{W}{\|W\|}$ et $\{W_i\}_{i \in I}$ une famille de H(T) recouvrant H(T). T minimise J sur D : $\Longleftrightarrow$ $\forall i \in I$ Reel $(W_i^*(A-\lambda I)^*(A-\lambda I)T) = 0$.

Nous montrons aisément que pour $\mathbb{K}^n = \mathbb{C}^n$ et les W_i vérifiant (4.3.19) nous avons bien $W_i = (\sqrt{-1})W_{i-1}$. La description de la base canonique est celle faisant suite au lemme 2.

D'où : T minimise J sur D

$\Longleftrightarrow \forall i \in I \quad W_i^* (A-\lambda I)^*(A-\lambda I)T = 0.$

Nous avons vu que $B = (A-\lambda I)^*(A-\lambda I)$ est une matrice définie positive, elle l'est même strictement si λ ne coïncide pas avec une valeur propre. Nous écrirons que :

$$T \text{ minimise } J \text{ sur } D \Longleftrightarrow \forall\ Z \in H(V), Z\ B\ T = 0$$

Nous pouvons donc extraire (n-1) équations linéairement indépendantes à n inconnues sur K. Elles sont complétées par les deux équations :

$$T^*T = 1$$

$$\text{et}\quad V^*T = 0$$

Un tel système n'admet pas de solutions dans le cas général. D'où V est unique au signe près.

Remarque : La convergence de cet algorithme n'impose pas de conditions particulières. Le vecteur V_0^1 initial pourra donc être choisi arbitrairement.

IV.3.4 - Détection des zéros numériques par la méthode de multiplication

Il nous est nécessaire de tester si :

$$f_i = \text{Reel}\ (W_i^*(A-\lambda I)^*(A-\lambda I)V)$$

est un zéro numérique. Ceci à la fois pour éviter une division par zéro ou une obtention d'une valeur de θ non significative, et pour poser un test d'arrêt des itérations lorsque toutes les valeurs f_i obtenues sont des zéros numériques.

La mise en place de la méthode de multiplication est aisée si nous constituons les matrices $C = k(A-\lambda I)$. Néanmoins, pour diminuer des réservations de place en mémoire, nous proposons l'adaptation suivante :

$$V' = k.V$$

$$W_i' = k.E_i - (V'.E_i)V'/k = k.W_i'$$

$$g' = \|W_i'\|^2 / k^2 = g$$

$$e' = J(W_i') - g\ J(V') = k^2.e$$

$$f' = \text{Reel}\ (W_i'^*(A-\lambda I)^*(A-\lambda I)V') = k^2.f$$

$$\theta' = \frac{e' - \sqrt{e'^2 + 4f'^2 g'}}{2f'.g'} = \theta$$

$$k.\frac{V' + \theta' W'_i}{\| V' + \theta' W'_i \|} = \frac{1}{1 + \theta'^2 g'} (V' + \theta' W'_i) = k.\frac{V + \theta W_i}{\| V + \theta W_i \|}$$

Lorsque le rapport signal/bruit du résultat numérique de f_i sera inférieur à 2, nous considérerons que la décroissance de J obtenue par une minimisation dans la direction W_i est négligeable et nous passerons cette étape. Si toutes les étapes au cours d'une itération ont été passées, alors nous considérerons avoir atteint le minimum.

L'estimation signal/bruit sera obtenue par le calcul de f' avec trois coefficients k possibles. Nous obtiendrons encore un résultat moyen $\bar{R}$ et un écart-type δ. Le test d'arrêt sera encore de la forme :

$$\text{Log}_{10} \frac{|\bar{R}|}{\delta} < \text{Log}_{10} 2 \simeq 0.3$$

IV.3.5 - Mise en oeuvre

Nous ne mettrons en oeuvre que le cas réel, le cas complexe ne présentant pas d'autres difficultés.

Nous avions deux soucis :

1) Le volume de calcul à chaque itération.
2) Une obtention maximale de précision.

1) Le volume de calcul pour une itération croît comme n^3 si nous appliquons brutalement l'algorithme de minimisation. Nous allons nous ramener à une croissance en n^2.

Si nous passons en revue les vecteurs nécessaires à une étape i, trois vecteurs seulement peuvent être utilisés à l'itération p :

$$R^p_{i-1} = (A-\lambda I)\, V^p_{i-1} \qquad (4.3.28)$$

$$W^p_i = E_i - (V^{p*}_{i-1}.E_i)V^p_{i-1} \qquad (4.3.29)$$

$$T^p_i = (A-\lambda I)\, W^p_i \qquad (4.3.30)$$

Le restant de l'algorithme peut se résumer à des produits scalaires ou à des opérations sur des scalaires.

Donc, seules les opérations (4.3.28), (4.3.29), (4.3.30) sont de l'ordre de n^2.

Supposons connu R^p_{i-1}.

V^p_{i-1}, R^p_{i-1}, E_i, W^p_i, T^p_i, sont des vecteurs colonnes.

Nous noterons leur jième coefficient par : $v^p_{i-1,j}$; $r^p_{i-1,j}$; $e_{i,j}$; $w^p_{i,j}$ et $t^p_{i,j}$.

Pour $\mathbb{K}^n = \mathbb{R}^n$, $V^* = V^t$ et nous pouvons alors obtenir directement $w^p_{i,j}$ par :

$$w^p_{i,j} = e_{i,j} - v^p_{i-1,i}.v^p_{i-1,j} \qquad (4.3.31)$$

ceci pour i,j = 1,n (nous rappelons que $e_{i,j} = 1$ pour $j = i$; $e_{i,j} = 0$ sinon).

Le calcul de T^p_i peut être décomposé en :

$$T^p_i = (A-\lambda I)E_i - [(V^p_{i-1})^t . E_i](A-\lambda I)V^p_{i-1}$$

soit :

$$t^p_{i,j} = a_{i,j} - \lambda e_{i,j} - v^p_{i-1,i} . r^p_{i-1,j} \qquad (4.3.32)$$

Nous pouvons déduire les coefficients e,f,g par :

$$g = W_i^t . W = 1 - v_i^2 \qquad (4.3.33)$$

$$e = J(W_i) - g\, J(V)$$

soit $$e = (T^p_i)^t . T^p_i - (1-v_i^2).(R^p_{i-1})^t . R^p_{i-1} \qquad (4.3.34)$$

et $$f = (V^p_{i-1})^t (A- I)^t (A- I) W_i$$

soit $$f = (R^p_{i-1})^t . T^p_i \qquad (4.3.35)$$

L'expression de θ reste inchangée, soit :

$$\theta = \frac{e - \sqrt{e^2+4f^2g}}{2\, f\, g} \qquad (4.3.36)$$

Nous pouvons déduire $||V^p_{i-1} + \theta_i W^p_i||$ par :

$$||V^p_{i-1}+\theta_i W^p_i|| = \sqrt{1 +\theta^2_i(1-(v^p_{i-1})^2)} \qquad (4.3.37)$$

et R^p_i par :

$$R^p_i = (A-\lambda I)\ \left(\frac{V^p_{i-1}+\theta_i W^p_i}{||V^p_{i-1}+\theta_i W^p_i||}\right) \qquad (4.3.38)$$

ou :

$$R^p_i = \frac{1}{\sqrt{1+\theta^2_i(1-(v^p_{i-1})^2)}}(R^p_{i-1}+\theta_i T^p_i) \qquad (4.3.39)$$

Pour finir, $J(V^p_i)$ sera déduit de :

$$J(V^p_i) = (R^p_i)^t.(R^p_i) \qquad (4.3.40)$$

2) L'expression de θ pour $J(V^p_i)$ tendant vers zéro devient rapidement inadéquate informatiquement. En effet, dans (4.3.36), le terme en $4f^2g$ est rapidement négligeable devant e^2, d'où obtention de résultats numériques de θ nuls.

Pour $4f^2g < \varepsilon$ avec $\varepsilon << e^2$, nous pouvons écrire :

$$\theta = \frac{e - e\sqrt{1+\frac{4f^2g}{e^2}}}{2\,f\,g} \simeq \frac{-\frac{2f^2g}{e}}{2\,f\,g} = \frac{-f}{e} \qquad (4.3.41)$$

Pour éviter un test à chaque étape, nous pourrons appliquer la formulation (4.3.41) dès que $J(V^p) < \varepsilon$.

Nous éviterons les conséquences d'approximations successives en renormant V^p après chaque itération, en calculant R^p_0 par (4.3.28) et $J(V^p_0)$ par (4.3.40).

Chaque étape ainsi définie nécessite un volume de calculs croissant comme n. Puisque nous effectuons n étapes pour une itération, nous aurons en définitive un volume d'opérations croissant comme n^2. Ceci est inchangé par la normation supplémentaire à chaque itération ou par l'étape supplémentaire de minimisation dans la direction de S définie en (4.3.25).

La mise en place de la méthode de multiplication (§ IV.3.4) sera une combinaison des formules (4.3.31) à (4.3.41) et de celles données au § IV.3.4. Les trois coefficients k_1, k_2, k_3 employés seront ceux du § II.4.2, soit : $k_1 = 1.$, $k_2 = 3.$, $k_3 = 5.$

IV.3.6 - Notice d'utilisation du programme MINF2

Objet

Amélioration du vecteur propre.

Appel

```
CALL MINF2(A,V,VAL,IA,N,R,RM2,T,S,IARG)
```

Arguments d'entrée

A Tableau bidimensionné dont la première dimension est IA et la seconde au moins égale à N. A contient la matrice dont nous cherchons le vecteur propre.

V,R,S,T sont des tableaux bidimensionnés dont la première dimension est IA et la deuxième 3.

V V(.,1) contient le vecteur propre initial.
Si IARG = 1 V est initialisé par le programme à V(I,1) = 1.
pour I = 1,N

VAL Valeur propre associée.

IA Première dimension des tableaux A,V,R,S,T.

N Ordre de la matrice.

IARG 0 le vecteur propre initial est donné par l'utilisateur.
1 il est calculé par le programme.

Arguments de sortie

V V(.,1) contient le vecteur propre amélioré et normé.

R R(.,1) contient le résidu défini par $R = (A - \lambda I)V$

RM2 Norme Euclidienne du résidu au carré.

Tableaux de travail

S,T

IV.3.7 - Programme FORTRAN MINF2

```
      SUBROUTINE MINF2(A,V,VAL,IA,N,R,RM2,T,V0,IARG)                    00000320
      DIMENSION T(IA,1),V(IA,1),R(IA,1),A(IA,1),RN2(3),IMUL(3),         00000330
     1 IMUL2(3),F(3),VALK(3),WX(3),V0(IA,1)                             00000340
      LOGICAL RETOUR,PR,ACCE                                            00000350
      COMMON/WXTST/CMIN,CMAX                                            00000360
      DATA IMUL/1,3,5/                                                  00000370
      DATA IMUL2/1,9,25/                                                00000380
C                                                                       00000390
C     SI IARG=1 ON PART D'UN VECTEUR PROPRE QUELCONQUE                  00000400
C                                                                       00000410
      ACCE=.FALSE.                                                      00000420
      EPS=1.E-05                                                        00000430
      PR=.FALSE.                                                        00000440
      NPR=1000                                                          00000450
      NITER=500                                                         00000460
      CMIN=1.0                                                          00000470
      CMAX=15                                                           00000480
      IFLAG=0                                                           00000490
      ITER=0                                                            00000500
      DO 11 K=1,3                                                       00000510
11    VALK(K)=VAL*IMUL(K)                                               00000520
      IF(IARG.EQ.0)GOTO 1                                               00000530
C     NITER=2000                                                        00000540
      NITER=2000                                                        00000550
      DO 2 I=1,N                                                        00000560
2     V(I,1)=1.                                                         00000570
1     CONTINUE                                                          00000580
C                                                                       00000590
C     ON NORME LE VECTEUR PROPRE                                        00000600
C                                                                       00000610
      SOM2=SQRT(PROSCA(V,V,N))                                          00000620
      DO 3 I=1,N                                                        00000630
      V(I,1)=(V(I,1)/SOM2)                                              00000640
      V(I,2)=V(I,1)*IMUL(2)                                             00000650
      V(I,3)=V(I,1)*IMUL(3)                                             00000660
      DO 40 K=1,3                                                       00000670
40    V0(I,K)=V(I,K)                                                    00000680
3     CONTINUE                                                          00000690
C                                                                       00000700
C     CALCUL DU RESIDU                                                  00000710
C                                                                       00000720
      DO 4 K=1,3                                                        00000730
      DO 5 I=1,N                                                        00000740
      SM=0                                                              00000750
      DO 6 J=1,N                                                        00000760
6     SM=SM+A(I,J)*V(J,K)                                               00000770
5     R(I,K)=SM-VAL*V(I,K)                                              00000780
4     RN2(K)=PROSCA(R(1,K),R(1,K),N)                                    00000790
C     PRINT *,RN2(1),RN2(2),RN2(3)                                      00000800
C                                                                       00000810
C     BOUCLE SUR LES ITERATIONS                                         00000820
C                                                                       00000830
7     CONTINUE                                                          00000840
      ITER=ITER+1                                                       00000850
      IF(ITER.GE.NPR)PR=.TRUE.                                          00000860
      NTEST=0                                                           00000870
      IF(RN2(1).LT.EPS)ACCE=.TRUE.                                      00000880
```

```
                DO 8 I=1,N                                                      00000890
                DO 9 K=1,3                                                      00000900
60              DO 10 J=1,N                                                     00000910
          10    T(J,K)=IMUL(K)*A(J,I)-V(I,K)*R(J,K)/IMUL(K)                     00000920
                T(I,K)=T(I,K)-VALK(K)                                           00000930
                F(K)=PROSCA(R(1,K),T(1,K),N)                                    00000940
          9     WX(K)=F(K)/IMUL2(K)                                             00000950
65        C     IF(PR)PRINT *,F(1),F(2),F(3),WX(1),WX(2),WX(3)                  00000960
          C                                                                     00000970
          C     TEST DE F                                                       00000980
          C                                                                     00000990
                CALL WXTEST(WX,3,C,RM,RETOUR)                                   00001000
70              IF(RETOUR)GOTO 12                                               00001010
                IF(RM.EQ.0)GOTO 12                                              00001020
                DO 13 K=1,3                                                     00001030
                TT=PROSCA(T(1,K),T(1,K),N)                                      00001040
                G=1-V(I,K)*V(I,K)/IMUL2(K)                                      00001050
75              E=TT-RN2(K)*G                                                   00001060
                IF(ACCE)GOTO 50                                                 00001070
                DELTA=E*E+4*F(K)*F(K)*G                                         00001080
                THETA=(E-SQRT(DELTA))/(2*F(K)*G)                                00001090
                GOTO 60                                                         00001100
80        50    THETA=-F(K)/E                                                   00001110
          60    CONTINUE                                                        00001120
                VN2=1+THETA*THETA*G                                             00001130
                FI=THETA*(THETA*E+2*F(K))/VN2                                   00001140
          C     IF(PR)PRINT *,TT,G,E,DELTA,THETA,VN2,FI                         00001150
85              VN=SQRT(VN2)                                                    00001160
                RN2(K)=RN2(K)+FI                                                00001170
                DO 14 J=1,N                                                     00001180
                R(J,K)=R(J,K)+THETA*T(J,K)                                      00001190
          14    T(J,K)=-THETA*V(I,K)*V(J,K)/IMUL(K)                             00001200
90              T(I,K)=T(I,K)+THETA*IMUL(K)                                     00001210
          C                                                                     00001220
          C     ON RENORME A IMUL(K)                                            00001230
          C                                                                     00001240
                DO 15 J=1,N                                                     00001250
95              R(J,K)=R(J,K)/VN                                                00001260
          15    V(J,K)=(V(J,K)+T(J,K))/VN                                       00001270
          13    CONTINUE                                                        00001280
                GOTO 8                                                          00001290
          12    NTEST=NTEST+1                                                   00001300
100       8     CONTINUE                                                        00001310
          C                                                                     00001320
          C     SI F N'EST NUL POUR AUCUNE COMPOSANTE ON CONTINUE A ITERER      00001330
          C                                                                     00001340
                DO 71 K=1,3                                                     00001350
105       71    WX(K)=RN2(K)/IMUL2(K)                                           00001360
                CALL WXTEST(WX,3,C,RM,RETOUR)                                   00001370
          C     PRINT *,ITER,RM,C                                               00001380
                IF(RETOUR) GOTO 17                                              00001390
                IF(ITER.GT.NITER)GOTO 16                                        00001400
110             IF(NTEST.EQ.N)GOTO 17                                           00001410
          C                                                                     00001420
          C     ON MINIMISE DANS LA DIRECTION DE S=(V.V0)V-V0                    00001430
          C                                                                     00001440
                DO 30 K=1,3                                                     00001450
```

```
      F(K)=PROSCA(V(1,K),V0(1,K),N)/IMUL2(K)                            00001460
      DO 42 I=1,N                                                       00001470
   42 V0(I,K)=F(K)*V(I,K)-V0(I,K)                                       00001480
      DO 31 I=1,N                                                       00001490
      T(I,K)=0.                                                         00001500
      DO 32 J=1,N                                                       00001510
   32 T(I,K)=T(I,K)+A(I,J)*V0(J,K)                                      00001520
   31 T(I,K)=T(I,K)-VAL*V0(I,K)                                         00001530
      F(K)=PROSCA(R(1,K),T(1,K),N)                                      00001540
   30 WX(K)=F(K)/IMUL2(K)                                               00001550
C     VERIF=PROSCA(V,V0,N)                                              00001560
C     PRINT *,VERIF                                                     00001570
C                                                                       00001580
C     TEST DE F                                                         00001590
C                                                                       00001600
      CALL WXTEST(WX,3,C,RM,RETOUR)                                     00001610
      IF(RETOUR) GOTO 1                                                 00001620
      IF(RM.EQ.0.)GOTO 1                                                00001630
      DO 34 K=1,3                                                       00001640
      TT=PROSCA(T(1,K),T(1,K),N)                                        00001650
      G=PROSCA(V0(1,K),V0(1,K),N)/IMUL2(K)                              00001660
      E=TT-RN2(K)*G                                                     00001670
      IF(ACCE)GOTO 51                                                   00001680
      DELTA=E*E+4*F(K)*F(K)*G                                           00001690
      THETA=(E-SQRT(DELTA))/(2*F(K)*G)                                  00001700
      GOTO 61                                                           00001710
   51 THETA=-F(K)/E                                                     00001720
   61 CONTINUE                                                          00001730
      VN2=1+THETA*THETA*G                                               00001740
      VN=SQRT(VN2)                                                      00001750
      DO 35 J=1,N                                                       00001760
   35 V(J,K)=(V(J,K)+THETA*V0(J,K))/VN                                  00001770
   34 CONTINUE                                                          00001780
C     IF(PR)VERIF=PROSCA(V,V,N)                                         00001790
C     IF(PR)PRINT *,VERIF                                               00001800
      GOTO 1                                                            00001810
   16 PRINT18,ITER                                                      00001820
   18 FORMAT(32H1 NON CONVERGENCE OBTENUE APRES,I5,11H ITERATIONS)       00001830
      IARG=-1                                                           00001840
   17 CONTINUE                                                          00001850
      DO 19 K=1,3                                                       00001860
   19 WX(K)=RN2(K)/IMUL2(K)                                             00001870
      CALL WXTEST(WX,3,C,RM2,RETOUR)                                    00001880
C     IF(RETOUR)RM2=0.                                                  00001890
      DO 21 I=1,N                                                       00001900
      DO 20 K=1,3                                                       00001910
      V(I,K)=V(I,K)/IMUL(K)                                             00001920
      R(I,K)=R(I,K)/IMUL(K)                                             00001930
      F(K)=R(I,K)                                                       00001940
   20 WX(K)=V(I,K)                                                      00001950
      CALL WXTEST(WX,3,T(I,1),RM,RETOUR)                                00001960
   21 CALL WXTEST(F ,3,T(I,2),RM,RETOUR)                                00001970
      RETURN                                                            00001980
      END                                                               00001990
```

```
      1              FUNCTION PROSCA(A,B,N)                                00002000
                     DIMENSION A(1),B(1)                                   00002010
                     PROSCA=0.                                             00002020
                     DO 1 I=1,N                                            00002030
      5              PROSCA=PROSCA+A(I)*B(I)                               00002040
                 1   CONTINUE                                             00002050
                     RETURN                                                00002060
                     END                                                   00002070
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 PROSCA

VARIABLES	SN	TYPE	RELOCATION						
0	A	REAL	ARRAY	F.P.	0	B	REAL	ARRAY	F.P.
16	I	INTEGER			0	N	INTEGER		F.P.
15	PROSCA	REAL							

STATEMENT LABELS
0 1

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
12	1	I	4 6	2B	INSTACK

STATISTICS
PROGRAM LENGTH 17B 15
55000B SCM USED

```
      1            DIMENSION A(20,20),V(20,3),R(20,3),T(20,3),S(20,3)          00000070
                   N=17                                                        00000080
                   IA=20                                                       00000090
                   DO3 I=1,N                                                   00000100
      5            DO 1 J=1,N                                                  00000110
                1  A(I,J)=0.                                                   00000120
                3  CONTINUE                                                    00000130
                   DO 2 I=1,N                                                  00000140
                   DO 20 J=I,N                                                 00000150
     10        20  A(I,J)=N-J+1                                               00000160
                   IF(I.EQ.1)GOTO 2                                            00000170
                   A(I,I-1)=N-I+1                                              00000180
                2  PRINT 21,(A(I,J),J=1,N)                                    00000190
                21 FORMAT(20F4.0)                                             00000200
     15             IARG=1                                                    00000210
                    VAL=1.                                                    00000220
                   CALL MINF2(A,V,VAL,IA,N,R,RM2,T,S,IARG)                    00000230
                   PRINT 5                                                    00000232
                5  FORMAT(1H1)                                                00000234
     20            DO 4 I=1,N                                                 00000240
                4  PRINT 6,V(I,1)                                             00000250
                6  FORMAT(/2X,   3X,E22.15)                                   00000260
                   RM=SQRT(RM2)                                               00000270
                   PRINT 12,RM                                                00000280
     25        12  FORMAT(3X,18HNORME DU RESIDU = ,E22.15)                    00000290
                   STOP                                                       00000300
                   END                                                        00000310
```

CARD NR. SEVERITY DETAILS DIAGNOSIS OF PROBLEM

1 I *PROGRAM START. (INPUT,OUTPUT)* ASSUMED WHEN HEADER STATEMENT IS DEFECTIVE OR OMITTED.

SYMBOLIC REFERENCE MAP (R=1)

VARIABLES	SN	TYPE	RELOCATION					
232	A	REAL	ARRAY	224	I	INTEGER		
223	IA	INTEGER		226	IARG	INTEGER		
225	J	INTEGER		222	N	INTEGER		
1146	R	REAL	ARRAY	231	RM	REAL		
230	RM2	REAL		1336	S	REAL	ARRAY	
1242	T	REAL	ARRAY	1052	V	REAL	ARRAY	
227	VAL	REAL						

FILE NAMES	MODE			
0	INPUT	20	OUTPUT	FMT

EXTERNALS	TYPE	ARGS			
MINF2		10	SQRT	REAL	1 LIBRARY

IV.3.8 - Résultats numériques

Nous testons toujours sur la matrice de II.1.6, mais cette fois-ci de dimensions 17x17. Nous voulions comparer nos résultats sur le vecteur propre associé à la valeur propre $\lambda = 1$. Il nous fallait donc des dimensions impaires. Ce vecteur propre a la propriété de posséder un coefficient sur deux nul.

Nous comparerons nos résultats avec ceux obtenus par le programme AEQI d'une bibliothèque mathématique. Les coefficients nuls du vecteur nous serviront de point de comparaison, sachant qu'il n'est pas possible d'obtenir une valeur 0. pour ces coefficients, mais une valeur qui soit la plus petite possible.

Après cette comparaison nous fournissons deux résultats supplémentaires montrant que le résidu normé n'est pas un bon moyen de contrôle pour la validité des couples valeurs propres-vecteurs propres. Les résultats associés à la valeur propre 17.15222... ont une bonne précision, mais un résidu normé important. Nous donnons ensuite un exemple d'une fausse valeur propre complexe avec un vecteur propre calculé faussement complexe. Le résultat du résidu normé semble pourtant être meilleur. Ceci montre bien que valeurs propres et vecteurs propres doivent être améliorés séparément.

Vecteurs propres associés à λ

-.861016233857091E-07	-.861016259933183E-07
-.594410518528107E-14	-.995405415217085E-15
.137762600073752E-05	.137762603537255E-05
.735839802656077E-13	-.191530344037181E-15
-.192867640687144E-04	-.192867644974019E-04
-.898992206165809E-12	-.243906510563242E-14
.231441169694227E-03	.231441173982182E-03
.897486237059304E-11	-.883434661686604E-14
-.231441170585123E-02	-.231441173977115E-02
-.717770861494010E-10	.715402179010811E-13
.185152937184984E-01	.185152939180039E-01
.430686355619545E-09	-.382737189927352E-12
-.111091762741639E+00	-.111091763507533E+00
-.172292334578102E-08	.130104892577937E-11
.444367052689923E+00	.444367054029295E+00
.344641060879165E-08	-.203094485447387E-11
-.888734108827101E+00	-.888734108057431E+00
NORME DU RESIDU = .230222501402125E-11	NORME DU RESIDU = .247315559046416E-11
λ = 0.99999999612...	λ = 1.

Tableau X

```
      1              DIMENSION A(20,20),VP(20,2),VECT(20,20,2),ITYPE(20),C(20,20)      00000070
                     N=17                                                              00000080
                     IA=20                                                             00000090
                     DO3 I=1,N                                                         00000100
      5              DO 1 J=1,N                                                        00000110
                1    A(I,J)=0.                                                         00000120
                3    CONTINUE                                                          00000130
                     DO 2 I=1,N                                                        00000140
                     DO 20 J=I,N                                                       00000150
     10         20   A(I,J)=N-J+1                                                      00000160
                     IF(I.EQ.1)GOTO 2                                                  00000170
                     A(I,I-1)=N-I+1                                                    00000180
                2    PRINT 21,(A(I,J),J=1,N)                                           00000190
                     DO 30 I=1,N                                                       00000200
     15              DO 30 J=1,N                                                       00000210
                30   C(I,J)=A(I,J)                                                     00000220
                21   FORMAT(20F4.0)                                                    00000230
                      T=48.                                                            00000240
                     CALL AEQI(IA,N,A,T,VP,VP(1,2),VECT,VECT(1,1,2),ITYPE)             00000250
     20         7     FORMAT(1H1,5X,2HNO,5X,13HPARTIE REELLE,21X,*PARTIE IMAGINAIRE*)  00000260
                     DO 4 I=1,N                                                        00000270
                     PRINT 7                                                           00000280
                     PRINT 8                                                           00000290
                     PRINT 6,VP(I,1),VP(I,2)                                           00000300
     25              PRINT 9                                                           00000310
                     DO 5 J=1,N                                                        00000320
                5    PRINT 6,VECT(J,I,1),VECT(J,I,2)                                   00000330
                     R=RN(C,IA,N,VECT,I,VP)                                            00000340
                4    PRINT 10,R                                                        00000350
     30         8    FORMAT(/5X,*VALEUR PROPRE*)                                       00000360
                9    FORMAT(/5X,*VECTEUR PROPRE*)                                      00000370
                10   FORMAT(/5X,*RESIDU NORME = *,E22.15)                              00000380
                6    FORMAT(/2X,   3X,E22.15,6X,   14X,E22.15)                         00000390
                     STOP                                                              00000400
     35              END                                                               00000410
```

CARD NR. SEVERITY DETAILS DIAGNOSIS OF PROBLEM

1 I *PROGRAM START. (INPUT,OUTPUT)* ASSUMED WHEN HEADER STATEMENT IS DEFECTIVE OR OMITTED.

SYMBOLIC REFERENCE MAP (R=1)

VARIABLES	SN	TYPE	RELOCATION				
322	A	REAL	ARRAY	2676	C	REAL	ARRAY
316	I	INTEGER		315	IA	INTEGER	
2652	ITYPE	INTEGER	ARRAY	317	J	INTEGER	
314	N	INTEGER		321	R	REAL	
320	T	REAL		1212	VECT	REAL	ARRAY
1142	VP	REAL	ARRAY				

```
      1              FUNCTION RN(A,IA,N,VECT,I,VP)                                    00000420
                     DIMENSION A(IA,1),VECT(IA,IA,2),VP(IA,2)                         00000430
                     TOT=0.                                                           00000440
                     DO 1 IS=1,2                                                      00000450
      5              IF(VP(I,IS).EQ.0.)GOTO 1                                         00000460
                     DO 2 K=1,N                                                       00000470
                     SOM=0.                                                           00000480
                     DO 3 J=1,N                                                       00000490
                3    SOM=A(K,J)*VECT(J,I,IS)+SOM                                      00000500
     10              IF(IS.EQ.2)GOTO 4                                                00000510
                     SOM=SOM-VP(I,1)*VECT(K,I,1)+VP(I,2)*VECT(K,I,2)                  00000520
                     GOTO 5                                                           00000530
                4    SOM=SOM-VP(I,2)*VECT(K,I,1)-VECT(K,I,2)*VP(I,1)                  00000540
                5    SOM=SOM*SOM                                                      00000550
     15              TOT=TOT+SOM                                                      00000555
                2    CONTINUE                                                         00000560
                 1   CONTINUE                                                         00000580
                     RN=SQRT(TOT)                                                     00000590
                     RETURN                                                           00000600
     20              END                                                              00000610
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4 RN

VARIABLES	SN	TYPE		RELOCATION						
0	A	REAL	ARRAY	F.P.	0	I	INTEGER		F.P.	
0	IA	INTEGER		F.P.	115	IS	INTEGER			
120	J	INTEGER			116	K	INTEGER			
0	N	INTEGER		F.P.	113	RN	REAL			
117	SOM	REAL			114	TOT	REAL			
0	VECT	REAL	ARRAY	F.P.	0	VP	REAL	ARRAY	F.P.	

EXTERNALS	TYPE	ARGS
SQRT	REAL	1 LIBRARY

STATEMENT LABELS

70	1	0	2	0	3		
56	4	64	5				

LOOPS	LABEL	INDEX	FROM-TO	LENGTH	PROPERTIES
30	1	IS	4 17	45B	NOT INNER
40	2	K	6 16	30B	NOT INNER
44	3	J	8 9	3B	INSTACK

STATISTICS
PROGRAM LENGTH 134B 92
55000B SCM USED

Tableau XI

NO	PARTIE REELLE	PARTIE IMAGINAIRE
VALEUR PROPRE		
	.171522110771268E+02	0.
VECTEUR PROPRE		
	.512392038386665E+00	0.
	.482518803031699E+00	0.
	-.235845429554151E-01	0.
	-.444183232539906E+00	0.
	-.399036468053641E+00	0.
	-.391167782064326E-01	0.
	.242336987325192E+00	0.
	.253294616758559E+00	0.
	.972410048962624E-01	0.
	-.413354471445342E-01	0.
	-.842799158613237E-01	0.
	-.624968323747073E-01	0.
	-.293712880942744E-01	0.
	-.944059528559543E-02	0.
	-.204063110068818E-02	0.
	-.270456000757612E-03	0.
	-.167442091658157E-04	0.
RESIDU NORME =	.416510805074660E-10	

Tableau XII

NO	PARTIE REELLE	PARTIE IMAGINAIRE
VALEUR PROPRE		
	.999999996127556E+00	0.
VECTEUR PROPRE		
	.861016244986223E-07	0.
	.567227978747851E-14	0.
	-.137762599953881E-05	0.
	-.745842149338803E-13	0.
	.192867640665359E-04	0.
	.896042139174467E-12	0.
	-.231441169692021E-03	0.
	-.896300569360370E-11	0.
	.231441170588403E-02	0.
	.717000572886599E-10	0.
	-.185152937187736E-01	0.
	-.430195275849372E-09	0.
	.111091762742835E+00	0.
	.172078432030619E-08	0.
	-.444367052692133E+00	0.
	-.344156746470663E-08	0.
	.888734108825840E+00	0.

RESIDU NORME = .246868432450098E-13

Tableau XIII

NO	PARTIE REELLE	PARTIE IMAGINAIRE
VALEUR PROPRE		
	.528172598828969E-01	.137738130539893E-01
VECTEUR PROPRE		
	-.541451285735941E-13	-.704647517300820E-15
	-.127081371924536E-12	.271950753281825E-13
	.419862560550354E-12	-.184722331407396E-12
	-.577132552504540E-11	-.232013389198306E-11
	.316407445791134E-10	.805517701045390E-10
	.565807302396368E-09	-.101276934782509E-08
	-.159785751039903E-07	.417539494287014E-08
	.189138168243530E-06	.825557610635052E-07
	-.905905105148395E-06	-.198545731320795E-05
	-.927914341993560E-05	.237217998594068E-04
	.247457847692468E-03	-.191631830926826E-03
	-.286964259039427E-02	.110555001404760E-02
	.218857532133361E-01	-.451259873169441E-02
	-.116429804931054E+00	.123375995514006E-01
	.422074082683388E+00	-.199332245173330E-01
	-.947182740117086E+00	.137738130539893E-01
	.100000000000000E+01	0.

RESIDU NORME = .391279113429125E-11

IV.4 - CONCLUSIONS

Nous pouvons dire qu'il n'existe pas d'algorithme d'amélioration de la précision efficace sans contrôle de l'erreur numérique propagée par cet algorithme. Les limites de l'amélioration de la précision sont fonctions de la stabilité numérique de l'algorithme employé.

Si le conditionnement d'une matrice est une constante quel que soit l'algorithme employé, la propagation des erreurs d'arrondi ou de troncature est dépendante de l'algorithme.

Nous avons vu deux méthodes d'amélioration de la valeur propre. Dans la première, nous minimisons la fonction $f(\lambda) = |\det(A - \lambda I)|^2$ à partir d'une solution approchée. Cet algorithme nécessite le calcul du déterminant. L'algorithme de Gauss employé nous paraît peu efficace, d'autres algorithmes existent, mais nécessitent un grand nombre d'opérations élémèntaires.

En attendant de trouver un algorithme de calcul du déterminant plus adéquat, nous proposons donc l'utilisation du polynôme caractéristique. Cette seconde méthode semble très efficace sur l'exemple proposé. De plus, elle est moins restrictive sur le choix des solutions initiales, et peut être même appliquée avec des valeurs propres initiales totalement fausses, c'est-à-dire avec zéro chiffre décimal significatif, ou même encore avec des valeurs propres mal séparées.

L'algorithme d'amélioration du vecteur propre semble très stable et ne nécessite aucune condition sur le vecteur initial. Ces capacités à améliorer un vecteur sont donc grandes, il peut même être employé comme nous l'avons fait pour calculer directement un vecteur propre.

V - CONCLUSIONS GENERALES ET EXTENSIONS

Les méthodes de permutation, perturbation ou de multiplication sont des moyens efficaces de contrôle du bruit numérique.

Nous avons vu aussi qu'elles permettent une mise en place d'un test d'arrêt optimal pour les méthodes itératives.

Nous avons abordé plus particulièrement le problème du calcul de la valeur propre dans le cas d'une matrice réelle quelconque.

La mise en place de la méthode de multiplication pour les algorithmes de valeurs ou vecteurs propres dans le cas de matrices tridiagonales, bandes, symétriques, complexes, etc... ne présente a priori pas plus de difficulté. L'attrait de cette méthode réside dans sa simplicité pour les problèmes d'algèbre linéaire.

D'un point de vue pratique, nous dirons que plus un algorithme présente des instabilités au bruit numérique, plus nos méthodes de contrôle semblent adaptées.

Dans ce cadre, le problème de la valeur propre généralisée : trouver un scalaire λ et un vecteur V tels que :

$$A\ V = \lambda B\ V \qquad (5.1)$$

(où A et B sont deux matrices nxn connues) est un problème intéressant !

En effet, inverser la matrice B pour résoudre ce problème peut engendrer de fortes instabilités |34,42| et en particulier lorsque A et B sont quelconques. Moler et Stewart |26| ont introduit l'algorithme QZ dérivé de l'algorithme QR. (Il existe toutefois un autre axe de recherche avec l'algorithme LZ |20|).

Une étude de la stabilité de l'algorithme QZ a déjà été effectuée en |2| en arithmétique exacte. Nous pensons avec certitude qu'en arithmétique à précision limitée, il faut s'attendre à une instabilité numérique plus importante encore avec cet algorithme QZ qu'avec l'algorithme QR.

Il faut prévoir ces prochaines années une complexité croissante des calculs accompagnant l'accroissement de puissance des gros ordinateurs. Par ailleurs, l'accroissement de mémoire de ces mêmes ordinateurs permet aussi d'accroître le dimensionnement du problème. Nous pensons donc que la validité des calculs deviendra rapidement la préoccupation essentielle du calcul scientifique.

Dans le IVe chapitre, nous avons abordé le problème de l'amélioration de la solution. Le problème qui se pose alors est souvent plus un problème économique lorsqu'il existe la possibilité de doubler la précision machine.

En effet, dans ce dernier cas, il est souvent plus efficace et moins coûteux de refaire les calculs en double précision que de procéder à une amélioration de la solution.

Cependant, l'augmentation de la précision a ses limites.

Dans une première démarche, nous avions cherché à améliorer des valeurs propres calculées en minimisant la fonction $f(\lambda) = |\det(A-\lambda I)|^2$. Malgré les résultats peu satisfaisants obtenus par cette méthode, nous continuons à penser qu'elle est intéressante. En effet, il est facile de généraliser et de construire par exemple la fonction $f(\lambda) = |\det(A-\lambda B)|^2$ dans le cas de la valeur propre généralisée.

Afin de rendre plus efficace cet algorithme, il sera nécessaire de faire une étude systématique de la stabilité des algorithmes de calcul du déterminant afin de choisir le plus adapté à notre problème. Les méthodes de contrôle du bruit numérique nous seront alors d'un grand secours.

Les résultats spectaculaires obtenus par l'utilisation du polynôme caractéristique sur notre cas test nous amènent à nous interroger sur l'instabilité réelle de cette méthode. Là encore, il serait nécessaire de tester un grand nombre de cas, et ce par tous les algorithmes de calcul du polynôme caractéristique existant.

Il semble moins aisé a priori de résoudre par cette voie le problème de la valeur propre généralisée. Cependant, il est toujours possible de trouver un polynôme admettant pour racine les valeurs propres généralisées solutions. Il ne nous est pas possible, dans l'état actuel de nos connaissances, de calculer ce polynôme sans calculer $B^{-1}.A$ (A, B définis en 5.1).

Dans la dernière partie, nous avons cherché un algorithme d'amélioration du vecteur propre, en minimisant la fonction $J(V) = ||(A-\lambda I)V||^2$. L'extension au problème généralisé ne présente guère de difficulté, nous utiliserons alors la fonction $J(V) = ||(A-\lambda B)V||^2$.

Nous regrettons que l'algorithme itératif trouvé n'ait pas une convergence quadratique. Mais nous pensons qu'il est certainement possible d'en accélérer la convergence. Nous trouverons des algorithmes d'accélération de la convergence notamment dans le livre de C.Brezinski |43|.

Nous voyons que le domaine d'application du contrôle du bruit numérique est vaste. La simplicité de mise en oeuvre de la méthode de multiplication en algèbre linéaire en fait un outil attractif pour le contrôle systématique du bruit numérique.

Nous voyons en cette méthode non seulement un moyen de contrôle, mais aussi une aide à la programmation ou au choix d'un algorithme stable.

BIBLIOGRAPHIE

|1| R. ALT
Etude statistique de l'erreur numérique d'affectation sur un ordinateur.
I.P. Report, n° 76, 5 february 1976.

|2| J.P. BERLIOZ
A propos de l'algorithme QZ.
RAIRO, Analyse numérique, vol. 13, n° 1, 1979, p. 21-30.

|3| P. BOIS et J. VIGNES
A Software for evaluating local accuracy in the Fourier transform.
IMACS, p. 141-150, vol. XXII, n° 2, july 1980.

|4| J. CEA
Optimisation, théorie et algorithme.
Dunod, Paris, 1971.

|5| CHAN, FELDMAN, PARLETT
A program for computing the condition numbers of Matrix Eigenvalues without computing eigenvectors.
ACM Transactions on Mathematical Software, Vol. 13, n° 2, june 1977, p. 186-203.

|6| F. CHATELIN-LABORDE
Sur les bornes d'erreur a posteriori pour les éléments propres linéaires.
Numer. Math., 32, 233-246, 1979.

|7| G. CHOQUET
Cours d'analyse - Tome 2 - Topologie.
2ème édition, Masson, Paris, 1973.

|8| D.K. FADDEEV and SOMINSKII, I.S
Problems in higher algebra.
2e éd., Moscow, 1949.
5e éd., Moscow, Gostekhizdat, 1954.

|9| J.G.F. FRANCIS
The QR transformation, parts I and II.
Comp. Jour., 4, p. 265-322, 1960-1961.

|10| F.R. GANTMACHER
The theory of Matrices. Vol. 1
Chelsa Publishing Company, 1959.

|11| B.S. GARBOW, J.W. BOYLE, J.J. DONGARRA, C.B. MOLER
Matrix Eigensystem Routines.
Eispack Guide Extension, Springer Verlag, 1977.

|12| S.I. GASS
Linear programming.
Mc Graw-Hill, New York, 1964.

|13| W.M. GENTLEMAN
Error Analysis of QR decomposition by Given'stransformations.
American Elsevies Publishing Company, Inc., 1975.

|14| GOLUB et WILKINSON
Ill conditioned eigensystems and the computation of the Jordan Cannonical Form.
Siam Review - Vol. 18 n° 4, october 1976.

|15| GRAD et BREBNER
Eigenvalues and eigenvectors of a real general matrix.
Comm. ACM 11, 820-830.

|16| K. HESSENBERG
Behandlung der linearen Eigenwert
Aufgaben mit Hilfe der Hamilton
Caylesjschen Gleichung. Darmstadt dissertation, 1940.

|17| A.S. HOUSEHOLDER
Unitary triangularization of a non-symmetric matrix.
J. Ass. Comp. Mach. 5, 339-342, 1958, b.

|18| A.S. HOUSEHOLDER
Generated error in rotational tridiagonalisation.
J. Ass. Comp. Mach, 5, p. 335-338, 1958, c.

|19| KATO
Perturbation theory for linear operators.
Springer Verlag 1966.

|20| L. KAUFMAN
TheLZ-Algorithm to solve the generalised eigenvalue problem.
SIAM Num. Anal, 11, p. 997-1025, 1974.

|21| B.H. KUBLANOVSKYA
On some algorithms for the solution of the complete problem of proper values.
J. Comput. Math and Math.-Phys., 1, p. 555-570, 1961.

|22| M. LA PORTE et J. VIGNES
Etude statistique des erreurs dans l'arithmétique des ordinateurs.
Application au contrôle des résultats d'algorithmes numériques.
Numerish Mathematik, 23, p. 63-72, 1974.

|23| M. LA PORTE et J. VIGNES
Evaluation de l'incertitude sur la solution d'un système linéaire.
Numerish Mathematik, 24, p. 39-47, 1975.

|24| MARTIN, PETERS et WILKINSON
The QR algorithm for real Hessenberg Matrices.
Numer. Math., 12, p. 349-368, 1968.

|25| M. MAILLE
Méthodes d'évaluation de la précision d'une mesure ou d'un calcul numérique.
Rapport LITP, Institut de Programmation, Université Pierre et Marie Curie, 1979.

|26| MOLER et STEWART
An algorithm for generalized Matrix eigenvalue problems.
SIAM, J. Numer. Anal., 10, 1973.

|27| OSBORNE
On pre-conditioning of matrices.
J. Assoc. Comput. Mach., 7, 338-345, 1960.

|28| B. PARLETT
Necessary and Sufficient conditions for convergence of the QR algorithm on the Hessenberg matrices.
Proceeding A.C.M. National Meeting, 1966.

|29| B. PARLETT
Canonical decomposition of Hessenberg matrices. Mathematics of Computation. Vol. XXI, n° 98, 1967.

|30| B. PARLETT et REINSCH
Balancing Matrix.
Numer. Math., 13, 293-304, 1969.

|31| H.H. ROSENBROCK
An automation method for finaling the greatest on least value of a function.
Computer Journal, Vol. 3, p. 175-184, 1960.

|32| RUTISHAUSER
Solution of Eigenvalue problems with the LR-Transformation.
Nat. Bur. Stan., Applied Mathematics Series, 49, p. 47-81, 1958.

|33| B.T. SMITH, J.M. BOYLE, J.J. DONGARRA, B.S. GARBOW, Y. IKEBE, V.C. KLEMA, C.B. MOLER
Matrix Eigensystem Routines.
Eispack Guide - 2nd ed., Springer Verlag, 1976.

|34| G.W. STEWART
On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$.
SIAM, J. Numer. Anal., 9, 1972.

|35| G.W. STEWART
Simultaneous Iterations for computing invariant subspaces of non hermitian matrices.
Num. Math., 25, 1976.

|36| J. VIGNES
Etude et mise en oeuvre d'algorithmes de recherche d'un extremum d'une fonction de plusieurs variables.
Thèse d'Etat, Faculté des Sciences, Paris, 1969.

|37| J. VIGNES
New Methods for evaluating the validity of the results of mathematical computations. Mathematics And Computers in Simulation. IMACS, p. 227-249, vol. XX, n° 4, 1978.

|38| J. VIGNES
Algorithmes numériques. Analyse et mise en oeuvre. Tomes 1 et 2. Technip, 1980.

|39| J.H. WILKINSON
Rounding errors in algebraïc processes.
Englewood Cliffs, N.J. Prentice Hall, 1963.

|40| J.H. WILKINSON
The algebraïc eigenvalue problem.
Clarendon Press. Oxford 1965.

|41| J.H. WILKINSON et C. REINSH
Linear Algebra - Tome II - Handbook for Automatic Computation.
Springer Verlag, 1971.

|42| J.H. WILKINSON
Some recent advances in numerical linear Algebra.
The State of the Art in numerical analysis, 1977.

|43| C. BREZINSKI
Algorithmes d'accélération de la convergence. Etude numérique.
Technip, 1978.